A Robot Operating System (ROS) Based Humanoid Robot Control

Ganesh Kumar Kalyani

A Thesis submitted to Middlesex University in fulfillment of the requirements for degree of

MASTER OF SCIENCE

Department of Design Engineering & Mathematics
Middlesex University, London

Supervisors
Dr. Vaibhav Gandhi
Dr. Zhijun Yang

November 2016

Tables of Contents

LIST OF FIGURES

LIST OF TABLES

Acknowledgement

The intensive period of my M.Sc. project work has been one of the best encounters I met with in my life. It has given me sufficient time to inculcate the research craving in me. I need to keep up a perfect balance, taking small and cautious steps moving steadily towards the objective. It is the consequence of sincere involvement extended by various people which enabled me to successfully complete this dissertation. I humbly express my true appreciation to every one of the individuals who were involved in my journey of Masters by Research. I earnestly hope that this thesis will be useful for anyone interested in Robotics research work.

I wish to pray and then would like to remember Almighty God ever for this great act of kindness and for showing me the light to fulfill my responsibility in writing up this thesis to the satisfaction and approval of Middlesex University and particularly the staff of Design Engineering and Mathematics Department.

My honest and sincere thanks are due to my Supervisors Dr. Vaibhav Gandhi and Dr. Zhijun Yang. I would like to remember their useful suggestions, master comments and valuable guidance which have been the heart and soul of this work from starting to finish. Their understanding, adaptability, genuine caring and concern, and encouragement shown to me during the dissertation process enabled me to take care of life and simultaneously likewise to acquire my Master Degree. They have been readily cooperating with me even after working hours and on holidays even. They had never discouraged me when they knew I needed to manage priorities. I appreciate their sense of judgment between research interests and personal pursuits. Further, I wish to thanks Mr. Nishant Singh, PhD researcher, Middlesex University for his support and help throughout.

Sincere thanks are also expressed to Middlesex University for having enabled me to carry out this research study.

Abstract

This thesis presents adapting techniques required to enhance the capability of a commercially available robot, namely, Robotis Bioloid Premium Humanoid Robot (BPHR). BeagleBone Black (BBB), the decision-making and implementing (intelligence providing) component, with multifunctional capabilities is used in this research. Robot operating System (ROS) and its libraries, as well as Python Script and its libraries have been developed and incorporated into the BBB. This fortified BBB intelligence providing component is then transplanted into the structure of the Robotis Bioloid humanoid robot, after removing the latter's original decision-making and implementing component (controller). Thus, this study revitalizes the Bioloid humanoid robot by converting it into a humanoid robot with multiple features that can be inherited using ROS. This is a first of its kind approach wherein ROS is used as the development framework in conjunction with the main BBB controller and the software impregnated with Python libraries is used to integrate robotic functions. A full ROS computation is developed and a high level Application Programming Interface (API) usable by software utilizing ROS services is also developed. In this revised two-legged-humanoid robot, USB2Dynamixel connector is used to operate the Dynamixel AX-12A actuators through the Wi-Fi interface of the fortified BBB. An accelerometer sensor supports balancing of the robot, and updates data to the BBB periodically. An Infrared (IR) sensor is used to detect obstacles. This dynamic model is used to actuate the motors mounted on the robot leg thereby resulting in a swing-stance period of the legs for a stable forward movement of the robot. The maximum walking speed of the robot is 0.5 feet/second, beyond this limit the robot becomes unstable. The angle at which the robot leans is governed by the feedback from the accelerometer sensor, which is 20 degrees. If the robot tilts beyond a specific degree, then it would come back to its standstill position and stop further movement. When the robot moves forward, the IR sensors sense obstacles in front of the robot. If an obstacle is detected within 35 cm, then the robot stops moving further. Implementation of ROS on top of the BBB (by replacing CM530 controller with the BBB) and using feedback controls from the accelerometer and IR sensor to control the two-legged robotic movement are the novelties of this work.

LIST OF ACRONYMS AND ABBREVIATIONS

AC	Alternate Current
ADC	Analog to Digital Converter
API	Application program interface
BBB	BeagleBone Black
BPHR	Bioloid Premium Humanoid Robot
I2C	Inter-Integrated Circuit
CAN	Control Area Network
CCD	Charge Coupled Device
CoG	Centre of Gravity
CoM	Centre of Mass
CoP	Centre of Pressure
CPG	Central Pattern Generator
CPU	Central Processing Unit
DC	Direct Current
DCS	Distributed Control System
DMS	Distance Measuring Sensor
DoF	Degrees of Freedom
F/T	Force / Torque
GND	Ground
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRF	Ground Reaction Force
GUI	Graphic user Interface
HAL	Hardware Abstraction Layer
IDE	Integrated Development Interface
IO	Input-output

IR	Infrared
KB	Kilo Byte
LAN	Local Area Network
LED	Light Emitting Diode
OS	Operating System
PC	Personal Computer
PD	Proportional Derivative
PWM	Pulse Width Modulation
RAM	Random Access Memory
RC	Remote Controller
ROS	Robotic Operating System
RRT	Rapidly Exploring Random Tree
RTX	Real Time Extension
Rx	Receive Connection
SCA	Serial Clock
SDA	Serial Data
SPI	Serial Peripheral Interface
TTL	Transistor-transistor logic
TV	Television
Tx	Transmit Connection
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
Wi-Fi	Wireless Fidelity
ZMP	Zero Moment Point

1. Introduction

1.1 Introduction

The term Robot means an automated, mechanized and an instrumented substitute-device for carrying out work that is usually carried out by humans. “Robotics” implies the branch of science and innovation that deals with the planning, development, utilization, and employment of robots and also computer framework for their operation, management, and information assessment [1]. Human body system includes bones as firm support system, cartilages that line the joints, muscles and ligaments that activate each part of the body. It is difficult to supplant this muscular-skeletal framework with the inert materials. Hence, machines could develop some equivalent human motions only [2].

Generally, a robot will have three types of components, namely, structural components, sensing components, and controlling and operating (deciding and intelligence providing) components [3]. The structural components are: i) manipulator or Rover: main body of the robot (links, joints etc.); ii) end-effector: usually the last component connected to the manipulator; (e.g., fore-hand with any needed device like gripper); and iii) actuators: muscles of the manipulator (servo motor, stepper motor, pneumatic and hydraulic cylinders etc. which moves the end-effector in the desired manner). Sensing and deciding and enforcing components are the other two important parts of robots.

There is noteworthy development in robotics and related scientific fields due to the access to modern sensors, powerful computing facilities, a wide range of active mechanical and electronic gadgets, in-depth and detail information about a wholesome robotic system, and different kinds of mechanical equipment’s etc. [4]. This research development on robots, especially at the university, is being undertaken with great zeal. In line with this development, the present work mainly focuses on robotic science and its development.

Repetitive and tedious tasks often involving precision and speed of control were the job of robots particularly in factories in the preliminary phase of robot's life. Human-robot symbiosis rapidly expanded and so human-robot interaction and autonomous movement attracted research [5]. A humanoid robot is built to visibly look like a person. A few of these robots may likely to have facial features for example, eyes and mouth to resemble a human. The study of design of a humanoid robot is either application based or research based. Humanoid robots are suitable for carrying out many human tasks such as helping people and carrying out simple jobs and also working in factory production line or dangerous environments [6].

Movement of a humanoid robot should be like that of a human, namely, exhibiting biped gait and legged locomotion. This is the biggest and most essential difference between a humanoid robot and other robots. The walking humanoid robot requires a system integration arrangement among its deciding and enforcing i.e., intelligence providing, sensing, and structural components. Preference is either for a centralized or a distributed control. When the setup is centralized, the main computer has access to all the information and hence all the calculations must be done by the main computer only and it is also not easy to increase the number of connections i.e., beyond that computers capacity [6]. In the distributed system, it is easily possible to provide many connections and attach several useful equipment's such as cameras, wireless LAN, control area network provision etc. However, computer or micro controller and motor controllers as well as provisions for connecting all of them are also required. In case Real Time Operating System (RTOS) is not used as a main controller but only a General-Purpose Operating System (GPOS), then real-time control program or the Real-Time Extension (RTX - which is a program readily available in the market) is also required. RTX can access hardware directly. For motion control, depending on the Degrees of Freedom (DOF), the maximum joint motor

controllers are possible to be worked out. For approximately 40 DOF, 15 joint motor controllers are easily provided. Here, the main controller receives sensor data quickly and forwards them to all the joint motor controllers [7].

Generally, the walking humanoid robot is configured with an autonomous controller and a motion controller. The autonomous controller takes care of the autonomous processing and image processing. The autonomous processor controls and directs the hardware, while image processing tools like Open CV[1] (Intel C/C++ libraries for computer vision) or Direct show[2] (Microsoft Multi Media Development Tool) are used for receiving images from the camera connected to the USB, while the program provided in the tools can carry out image processing [8, 9]. The motion controller may include a walking program and a special action program written in any suitable programming language such as C, C++, etc. The firmware program can be included in the motion controller, which if necessary, can be modified to suit the environment using a window note-book PC or other suitable devices.

For deciding whether a PC, Microcontroller and motor controller set-up is better or only a microcontroller and motor controller is enough the following points need consideration:

1. Powerful processor is used for providing Artificial Intelligence and overall control and simpler microprocessor as I/O managers are being used in walking humanoid robots.
2. Preferably, three-layer configuration is adopted:

    i. Top layer - Powerful PC as brain

[1] OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. The library is cross-platform and free for use under the open-source BSD license.

[2] The Microsoft *DirectShow* application programming interface (API) is a media-streaming architecture for Microsoft Windows. Using *DirectShow*, your applications can perform high-quality video and audio playback or capture. The *DirectShow* headers, libraries, SDK (Software Development Kit) tools, and samples are available in the Windows SDK.

ii. Middle layer - Multiple sub-system managers for motion control, sensor management, sensor processing etc.
iii. Lower layer - Number of cheap microprocessors for managing miscellaneous tasks

Ethernet or USB will be suitable for upper layer and RS232[3], I2C[4], etc. are used for lower level communication [10]

3. Microprocessor is plugged in to a USB on the Notebook and it will easily send and receive commands over serial port and all other higher brain programming code could run on the Notebook. The Notebook could issue commands by taking decisions and the other processor could control the motors to execute the required tasks [10].
4. Decisions on the use of processor is taken considering the following:

i. Processing Power-If controlling the motors is only needed (i.e. all processing is done on the external PC) a motor driver controller is enough. For basic on-board processing like transformation of coordinates, simple command loops and inverse kinematics, a simple microprocessor is sufficient. If automating the robot and implementing advanced features are desired, BeagleBone Black, Raspberry Pi or Intel Atom program is sufficient.

[3] In broadcast communications, RS-232 is a standard for serial correspondence transmission of data. It formally characterizes the signals interfacing between a Data Terminal Equipment (DTE), for example, a work station, and a Data Communication Equipment (DCE), for example, a modem.
[4] I2C is a serial protocol for two-wire interface to connect low-speed devices like microcontrollers, EEPROMs, A/D and D/A converters, I/O interfaces and other similar peripherals in embedded systems. Each I2C slave device needs an address.

ii. Programming – Basic[5], Python[6] or C/C++[7] language is required for BeagleBone Black. But wherever necessary, libraries are included in addition.

iii. Compatibility - The chosen processor should support (both number and type) the motors used [11].

The following points may generally be taken into consideration in assessing the requirement of software required for robot design:

1. ROS (as development framework) depends on the machine controller that integrates robotic functions [12].
2. Python API allows either to use the entire C++APIs from a remote machine or Python Modules must be created that can run remotely or on the robots [13].
3. Embedded software, running on the mother-board located in the head of the robot allows autonomous behaviour. Desktop software running on a computer located outside the robot, allows creation of new behaviours and the remote control of the robot [13].

The above theoretical considerations were carefully considered for evolving a ROS based humanoid robot for autonomous control, by combining two commercially available robots. The compatibility and suitability of both the replaced and replacing deciding and enforcing components were carried out. Thereafter, it was noticed that the substitution of the BeagleBone Black (BBB) deciding and enforcing multifunctional component (after strengthening it with

[5] BASIC (an acronym for Beginner's All-purpose Symbolic Instruction Code) is a family of general-purpose, high-level programming languages whose design philosophy emphasizes ease of use.
[6] Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java.
[7] C++ is a high-level programming language it adds object-oriented features to its predecessor C. C++ is one of the most popular programming language for graphical applications, such as those that run in Windows and Macintosh environments.

dynamic walking capacity) instead of or in place of the CM530 deciding and enforcing component of Bioloid Premium Humanoid Robot (BPHR) (with walking capacity alone) was workable and carried out in this study. Thus, the adapting strategy of implanting the fortified intelligent providing component of BBB in the structural body (without its original intelligent providing component) of BPHR is implemented successfully in this research project.

1.2 Rationale

With the advancement of robots and robotics as a discipline of study humans have been interested in building human-like robots (both in structure and function). This desire came true to some extent, with the advent of Humanoid Robots. There were further expectations that humanoid robot should help and take care of humans in their homes, hospitals and factory floors with mild and simple jobs. These advancements and reciprocations have long surpassed [5]. Now-a-days, robots are used for arduous tasks in mines, seas, under-water, space, and hazardous environments like nuclear plants etc. [5]. Apart from two-legged dynamic motion, even four–legged flexible motion to negotiate pipes, steps, slippery floors etc. are very much in demand [5]. Although there is a high demand the market is not yet producing such tough humanoids suitable for hazardous work. The traditional and established robotic platforms seem lacking in efficiency to meet such challenges and demands. Therefore, it is necessary to intertwine the mere dynamic walking and simple working capabilities of humanoid robots like Bioloid, KHR [7], Honda ASMIO [14], the WABOT series of Waseda University [15], and H6 and H7 of Tokyo University [16], with multifunctional capacities of BBB (and similar) deciding and enforcing (intelligent providing) component. Such innovation with autonomous computing hardware has paved way for the much-needed flexibility in the use of hardware and software integration for the betterment of robotics. Hence, this thesis research work is undertaken to remove the deciding and enforcing (intelligence providing)

components of BPHR and implant in its body, instead, the advanced multifunctional intelligence providing component of BBB after making necessary changes and including in addition ROS software in Python Script with libraries. This type of research is an imperative need of the hour and would become promising as it would be befitting and beneficial and thus, capable of setting the baseline for alternate robotics studies.

1.3 Aim and Objective

Biped humanoid robots and quadruped or hexapod legged mechanical but some-what more intelligent devices, recognized and referred to as robots, are in wide usage. It is the avowed aim and object of this research study to merge them together and develop a two-legged humanoid robot to have dynamic human-like stable walking with the ability of path-finding and potentially negotiating obstacles etc.

A ready-made humanoid robot of one firm capable of walking and performing some simple functions carried out by human beings is blended with another highly intelligent robotic device with multifunctional capacities resulting in a humanoid robot capable of performing all the functions of the above two robots. This research study is the first of its kind as it brings together the concepts of ROS and replaces the CM530 controller with BBB. The walking style adopted here is an attempt to provide humanoid with the ability of bipedal walking without falling down.

BBB controller is mounted in place of the built-in CM530 controller within the BPHR. Individual Dynamixel servos are accessed through BBB via USB2Dynamixel connector. To meet the challenges of keeping the robot upright and maintain its balance, a gyro/accelerometer sensor is used with an IR sensor for avoiding the obstacles while walking. ROS is implemented in the BBB and the walking of the BPHR is to be made performing using python programming

language with BBB. The sensors are used for obstacle avoidance and for providing stability while walking. The Wi-Fi in the BBB is enabled so that the humanoid robot can be controlled from a distance.

The python software is modified to run by using ROS alone. Various modes of walking such as slow, medium and fast. There is no plan of making any physical modifications to the robot. The humanoid robot is to walk with the python code making use of BBB sensors and avoid collision too.

1.4 Outline of the Thesis

The thesis is divided into five chapters.

Chapter 1: This chapter describes rationale, aims, goals and outline of the thesis.

Chapter 2: This chapter details the existing robotics technological advancements and presents in detail the basic knowledge and theoretical considerations involved. This chapter also formulates the scheme for adaptability of installing the decision-making and implementing component of BBB into the structure of biped walking BPHR. Further, this chapter also outlines the other considerations relating to robots such as their types, parts and modes of controls. The mechanics involved in deciding and implementing the various decisions that are taken have also been addressed in this chapter. The walking action performed by the robot including the electronic parts required, efficient way of programming, algorithms for various robot's actions and developing codes for controlling robots are put forth in detail. The desirable requirements including expected features and essential functions are also described.

Chapter 3: This chapter describes in detail the BBB decision-making and implementing (intelligence providing) component, also giving details about its versatile multi-function capability. This chapter also gives details about Robotis make BPHR, particularly its structural features such as manipulators, end-

effector, and actuators and also some details about the limited capability of the built-in CM530 controller.

Chapter 4: This chapter furnishes in detail the important aspects of research study undertaken and achieved in this thesis, particularly empowering the Bioloid, GP humanoid biped robot with multifunctional capabilities of the BBB's decision-making capacity. The BBB instills dynamic and stable walking capacity in the BPHR, which was previously lacking with the CM530 controller.

Chapter 5: This chapter concludes the thesis with the basic contributions listed. Future work is also discussed.

2.Literature Review

This chapter details the generic humanoid robot structure, and its autonomous working and control in recent years. After carrying out a literature survey, a suitable algorithm is to be selected for robot's walking and stability, and is to be implemented in the BeagleBone Black. This is to be achieved with the selected walking control trajectory being guided by the software, developed using Robot Operating System (ROS) and Python language and their libraries.

2.1 Basics of Robotics

2.1.1 Historical Development

From the Russian word 'paboTa (Rabota)' meaning work or labour, the Czechoslovakian author named Karel Capek developed the word 'Robots'. The robots are referred to the programmable machines used for performing manipulation or locomotion functions under automatic control. Most robots are inspired by nature adding to the field of bio-propelled ones. This has led to the development of another new branch of robotics called soft robotics. Currently, robots that function and act like humans also have the fondness to exist together with people as expected and symbiotically [2]. This is called emotional robotics and is the present attraction of industry and research.

Robots have come a long way, from the wooden model of a pigeon activated by a steam jet to the Hitachi's brain child "HIVIP Mk.1" intelligent robot; which is capable of understanding and performing tasks from line drawings, recognizing orientation of parts through computer vision, and automatic planning of sequence of motions required for the specified task (1970) [5]. There has been a phenomenal growth in robotic technology. Advanced Quadruped Robots and Human-Symbiotic Robots are being rapidly developed since 2000, creating even fear and panic in people culminating in Asimov's "Three laws Of Robotics" for the robots to be no danger for humans [17].

Robotics research activities, benefited from the fast improvement of technologies, have been dramatically increased in many new cutting edge fields, such as formal methods, computer vision, image processing and artificial intelligence [17].

2.1.2 Structural components

The three important structural components of a humanoid robot namely, manipulator, end-effector, and actuator are shown in Figure 1.

Sensing components are the sensors that obtain and measure information about the condition of the robot and external condition to effectively deal with the real-world. The third decision making enforcing components are: i) Processor-the brain (calculates motion and velocity of robot's joints); ii) Controllers-cerebellum of the brain (controls and correlates the motion of actuators); and iii) Software-Operating system (the programming platform providing for motion, task, and managing functions, like the tools and library for conveying and collection of routine information) [3]. Further, power connection is needed for robot to enable its actuators to function effectively. Power is mostly obtained from batteries or wall-mounted electrical plugs. Alternately, they may use either pressurized hydraulic fluid using pumps or pneumatically compressed air using air

Figure 1: Structural Components of a Humanoid Robot.

Source [7 & 18]

compressor and compressed air tanks. Wired electrical circuit interfaces all actuators, and powers every electrical motor and solenoids directly. In addition, the circuit operates the hydraulic equipment by controlling electrical valves which decide the pressurized liquid's circulation through the machine. For moving a hydraulic leg, for instance, the robot's controller would open the valve driving liquid from the liquid pump to a cylinder barrel joined with that leg. This pressurized liquid would move the piston forward thereby stretching the leg in the front direction. Robots utilize cylinders that can thrust in both directions to move their actuators in two opposite directions i.e., front and reverse. Robot's PC directs everything appended to the circuit. The PC operates all the essential motors and valves through microcontrollers and motor controllers to move the robot further [19].

A microcontroller can operate motor controllers in an assortment of ways: serial, I2C, PWM and R/C. Irrespective of the communication mode, controller's logic and the microcontroller has to use similar ground reference and a similar high level logic (which can be accomplished by utilizing the same V+ pin to operate and control both gadgets). A logic level shifter might be required if the gadgets don't share similar logic levels (3.3V and 5V for instance). Sensors can be interfaced with microcontrollers in the same manner as motor controllers. Sensors can be reached through the following kinds of communication: Digital, Analogue, Serial or I2C. Most communication technologies (e.g. ZigBee[8], Bluetooth[9]) adopt serial communication, so the same Rx (Receive Connection), Tx (Transmit Connection), GND (Ground) and V+ (Supply) links serves the purpose [20].

[8] ZigBee is a wireless technology developed as an open global standard to address the unique needs of low-cost, low-power wireless M2M networks. The ZigBee standard operates on the IEEE 802.15.4 physical radio specification and operates in unlicensed bands including 2.4 GHz, 900 MHz and 868 MHz.

[9] Bluetooth is a wireless technology standard for exchanging data over short distances (using short-wavelength UHF radio waves in the ISM band from 2.4 to 2.485 GHz) from fixed and mobile devices, and building personal area networks (PANs).

2.2 Control of Robot Functions

2.2.1 Remotely controlling a robot

Remotely controlled torpedoes developed during late 19th Century may probably be the first remotely operated vehicles or robots. The most effortless approach to remotely operate a vehicle is with a hand held controller physically linked with the vehicle through wires. The controller may have a Toggle Switch, knob, lever, joystick, or button for controlling the vehicle without using complex electronics. The motor and power source can be linked with operated with a switch for controlling the forward or backwards rotations.

At times a cable may be a better choice as there is no limitation of operation time since power is availed from the mains, there is no loss of signals, and it is simple without complicated electronics, etc. However, its disadvantages are: the distance may be limited, dragging the cable may lower the speed of robot, cable may get damaged etc. Wired computer control can be provided by installing a microcontroller in the vehicle and using a cable to connect it with a computer's I/O port (USB). More complex behaviors can be programmed, which may give onboard intelligence etc. but cost may increase because of the added electronics. Ethernet connection could also be made with cable connection and thus control could be via the Internet but may require complex programming. Wire-less radio frequencies, Bluetooth, and Wi-Fi connections may also be used for remote control purposes without using cables.

2.2.2 Semi-Autonomous Control

Often a joint control, in which some aspects are controlled by a human and others are done by the robot itself, can be the best option. For example, in an advanced submersible robot, fundamental mobility of the robot is directed by a human while an on-board processor reads and responds to streams running below the water so as to keep the robot stable. Human being gives additional directions if

any after getting a video feedback from the on-board camera. Water temperature, pressure, and other relevant data are also tracked by the on-board sensors. In case the communication gets lost, an autonomous program can bring the robot to the surface. The level of autonomy has to be decided on priority. Similarly, humans can best interact and intervene in the case of a semi-autonomous wheelchair.

2.2.3 Autonomous Control

Self-governing business and factory robots are currently engaged extensively carrying out works more economically, or with more noteworthy exactness and dependability than people. For having more autonomy, microcontroller must be used with its full potential and programming has to be done for the microcontroller to react to input from the robot's sensors. Pre-programming with no correction data from the surrounding, restricted input data from sensors, and lastly complicated sensor correction-information are the different types of self-governing control. Incorporating an assortment of sensors and code which permits the robot to decide without any other external input, the best move to be made in any noted circumstance, which indicates true autonomous control.

2.3 Decision-making & Implementing Mechanism

2.3.1 Electronic components

Main piece of electronic device that is essentially needed for all computations, decision making and communications is a microcontroller. This microcontroller can be seen in everyday products like TV sets, washing machines, telephones, watches, practically in every home appliance and electrical device. It is a computing device capable of executing programs, of course, not involving complex algorithms. It acts as the brain or center of the Robot. Hence it is less capable than a PC and further they send a little quantity of electrical power through its pins. So it won't have the capacity to work with huge load specifically. Arrangement of set of pins (electrical pulse communications) are there that can

be turned HIGH (1/ON) or LOW (0/OFF) through programming guidelines. They additionally read electrical pulses from sensors or different gadgets. Microcontrollers can be utilized to control other electrical gadgets, for example, actuators when attached with motor controllers. They additionally incorporate a voltage controller in their improvement sheets.

Continuous Voltage pulses (pulses that can have a full scope of qualities rather than only two very much characterized states like, 0 and 1 in Digital pulses) can likewise be measured by most present day microcontrollers with the help of Analog to Digital Converter (ADC). With the assistance of the ADC a microcontroller can allot a numerical amount to a series of continuous pulses which is neither high nor low. If it is noticed that a device to be connected provides a value which is proportional to some factor, say, temperature, force, position etc. then the microcontroller must be provided with an analog pin. Although microcontrollers appear to be limited in its capacity it can perform many complex functions if its pins are set HIGH and LOW in a logical way. However, because of its inherent resource and speed limitations it is not possible for a microcontroller to handle very complex algorithms like advanced vision processing and very complex programs.

The microcontroller is akin to a computer CPU (microprocessor) and its improvement board is similar to PC mother board. In advanced Robots with complex computing and vision algorithm etc. both a PC for overall control and a microcontroller for accessing motors are used.

Light and medium robots may only need a microcontroller while heavy ones may require a PC in addition as shown in Figure 2. A microcontroller might not have the capacity to provide power to electrical motors since its output can only furnish a meager quantity of electrical power. For a larger robot, more power will be required to run its motors.

The microcontrollers can accomplish more than the typical (usual) digital I/O, (essential calculation, fundamental arithmetic and decision making) if special hardware, for e.g., a motor driver is built into them. Most popular communication protocols like UART[10], SPI[11] and FC can be readily supported by many microcontrollers. This is profoundly valuable for corresponding with different gadgets like PCs, improved sensors, or different microcontrollers. Regardless of the possibility that it is conceivable to physically work orderly commands, committed inherent equipment which manages the particulars is particularly preferable. Such a course of action permits the microcontroller to focus on different assignments and gives room for cleaner programs.

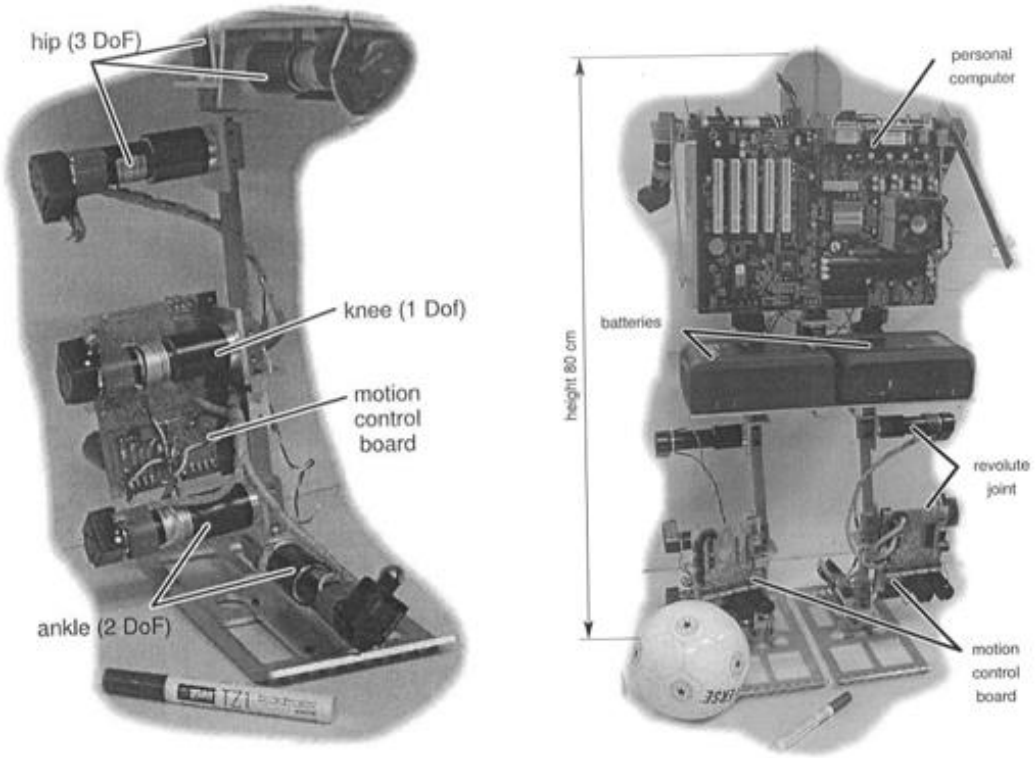

Figure 2: Heavy Robot Arduous Jobs

Source [17]

[10] A universal asynchronous receiver/transmitter (UART), is a PC equipment gadget that translates information between parallel and serial structures. UARTs are ordinarily utilized as a part of conjunction with correspondence principles, for example, TIA (once in the past EIA) RS-232, RS-422 or RS-485

[11] Serial Peripheral Interface (SPI) is an interface bus commonly used to send data between microcontrollers and small peripherals such as shift registers, sensors, and SD cards. It uses separate clock and data lines, along with a select line to choose the device to talk.

Next in importance is the Motor Controller which is an electronic device, an uncovered circuit board without being walled in an area that goes about as a middle of the road gadget between a microcontroller, a power supply or battery, and the motors. Since there are several types of motors as in Figure 3 like brushed DC (with or without gears) motors, brushless DC motors, linear actuators, Servo motors, unipolar or bipolar stepper motors etc. there are as many varieties of motor controllers as well. The speed and direction of these motors are decided by the microcontroller only, but the motors cannot be driven by the microcontrollers because of its restricted power. The motor controller provides necessary power at the required voltage and drives these motors. Both these controllers work together to make the motors move appropriately. Regular and easy communication technique such as UART or PWM is used by the microcontroller to give information the motor controller as to how to power the motors.

Motor controller's physical size depends on the size of motors. A motor controller smaller than the tip of a finger has enough current capacity to drive a mini sumo robot (robots attempting to push each other out of a small arena) while a large controller weighing several kilograms (including heat sink etc.) may be required for an unmanned aerial vehicle. The size thus depends on the amount of power the controller has to provide.

Selection of motor controllers depends on the current consumed by the motor which it is intended to control. The existing a motor attracts is associated to the

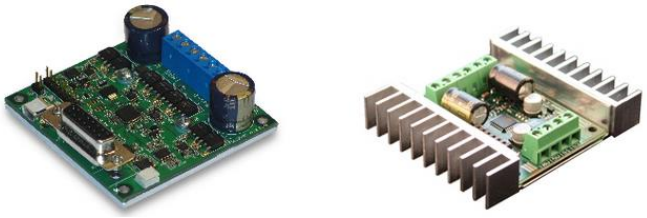

Figure 3: Types of Motor Controllers

Source [21]

torque it produces i.e., more current will be drawn to produce higher torque by a large motor and vice-versa.

Motors are connected to motor controllers which in turn are connected to the microcontrollers. All the sensors are also connected to the microcontroller. Under certain conditions one computer (cf. Figure 2) or two computers (cf. Figure 4) are also added in addition to microcontroller and motor controllers. Batteries are connected to motor controllers, microcontrollers and the main computers.

2.3.2 Programming Essentials

Programming a microcontroller is easy now-a-days due to current Integrated Development Environments (IDE) that utilize most recent languages, completely highlighted libraries that promptly cover all regular and rare activities and a few instantly usable codes for beginners with easy use. Different high level languages like C, C++, C#, processing (a variety of C++), Java, Python, .NET, BASIC etc. can be programmed into a microprocessor without much difficulty.

IDEs are likewise turning out to be much less complex as makers make graphical programming environments. Groupings requiring a few lines of code are diminished to a picture that can be associated with different other pictures to form

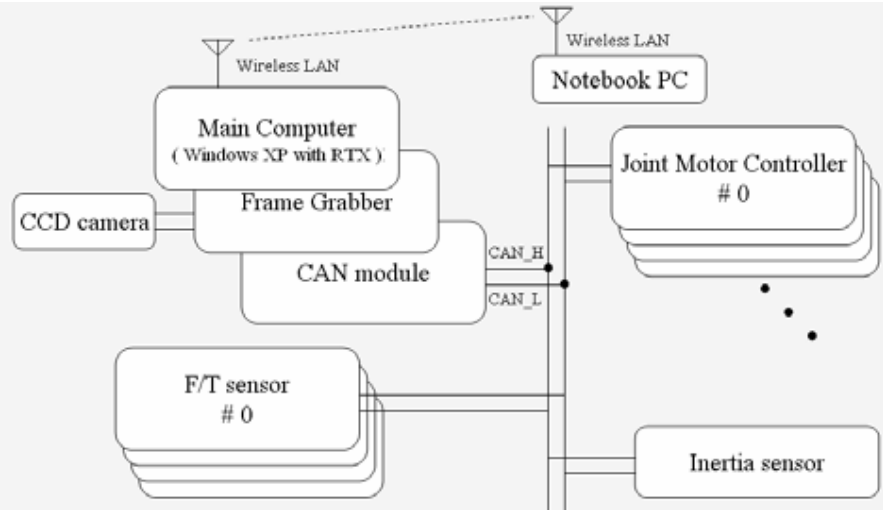

Figure 4: Two PCs with Other Controllers

Source [7]

code. If the user needs to control a motor in a robot, the user should mount the image which represents controlling a motor and should also specify the direction and speed at which the motor should run. Microcontroller Development Boards break out each of the pins utilized by the microcontroller and makes them simple to be accessed for fast circuit prototyping. Advantageous USB power and programming interfaces that connect appropriate to any advanced PC are also provided by the board. These Development Boards are the only circuit boards that provide microcontroller chips with all the necessary supporting hardware, for example, the voltage controller.

2.3.3 Python Language

2.3.3.1 Selection of programming language

Programming is the last step and without it the robot cannot function at all. Developing the software for a computer is a task which cannot be dealt with in this thesis. However, programming the microcontroller can be discussed.

The general types of languages and their main features are indicated below:

- Basic - This is one of the earliest languages which is still being used in some microcontrollers like Basic micro, BasicX, Parallax etc. in educational robots
- C/C++ - This is one of the well-known languages which gives high level functionality but at the same time keeping a good low level control.
- Java - It is more recent language than C and possesses plenty of safety features but not low level control. It is best suited for microcontrollers produced by Parallax Company.
- NET/C# -This Microsoft's proprietary language is utilized to develop as Netduino, FEZ Rhino etc.
- Processing - It is a variant of C++ having lots of simplifications to make programming easier in Arduino.

- Python - It is a most popular, easy to learn and enabling to place programs jointly very quick and efficiently scripting language. Hence the Python language is chosen in this research work.

2.3.3.2 Considerations in code writing

- Only limited, that is, manageable length of code essentially needed to each product must be developed along with building a library and creating a file system for easy tracing up of the required code.
- Everything, (that is, every line) within a code must be documented using comments particularly in robotics that too at the initial stages. Later on after acquiring enough knowledge and experience it will be enough to add comments to general sections of code.
- Different versions of code developed must be preserved without overwriting the same file so that when a later developed code does not compile it should be possible to revert back to a previously developed one instead of trying to rectify the later one. This correcting work may prove to be tedious and time consuming.
- Keep the robot always in a safe position so that it will not destroy itself or get destroyed in careless or erroneous trials.
- Power should be turned off when it is noticed that code is not carrying out correctly the expected functions.
- Converting a portion of the code into subroutines will make the usage of code easy to use if that section of code is recurring numerous times inside a program.

2.3.4 Decision-making and Enforcing Technique

Battery is connected to the main computer, all sub-controllers, all sensors, a Charge Coupled Device (CCD) camera and a frame grabber, the Controller Area Network (CAN) module, fans etc. The inputs from various sensors like

force/Torque Sensor, Inertia sensor, Distance Measuring Sensor (DMS), Thermometer, Pressure measuring sensor, Rate Gyro Acceleration Sensor, Infrared (IR) Sensor[12] etc. are received by the main computer which are forwarded to appropriate motor controllers through the main microcontrollers for operating the concerned actuators in the desired manner in response to the outside world condition. These enforcing instructions from the main computer emanate from out of the software included in them. Thus intelligence is imparted to the robots for deciding and enforcing functions.

2.3.5 Control Architecture

2.3.5.1 Main Controller

For light and medium robots, microcontroller will be the main controller whereas for heavy, multifunctional and humanoid robot one or even two PCs will be used as main controllers along with microcontroller and many motor controllers. When a number of peripheral interfaces, simple and quick programming environment and good graphic user interface (GUI) are required PC is the best choice as main controller.

2.3.5.2 Control Area Network (CAN) Protocol

When Computers are used as main controller along with microcontroller and many motor controllers, for receiving many kinds of data from various devices and to give orders by the main computer, there must be bus line for correspondence between the primary computer and other controllers (cf. Figure 4). For dealing with the number of sub-controller, fast communication is essential. The CAN protocol used in automobile industry with serial communication speed of 1 megabit per second is found suitable. It is very easy to

[12] An infrared sensor is an electronic instrument which is utilized to detect/assess the prevailing nature/qualities of its surroundings by either transmitting or potentially recognizing infrared radiation.

expand the number of additional sub-controllers since only two lines are needed in this protocol. All controllers connected to this bus line are able to send out and also collect data one and the same time [7].

2.3.5.3 Sub-controllers

Trunk roll controller (for preventing the slope of the trunk) due to the transform in the ground slope utilizing rate accelerometer, landing position controller (for modifying the position schedule to prevent unstable landing of the robot if the landing takes place before or after the prescribed time), damping controller and landing orientation controller (to modify joints angles prescribed by inverse Kinematics by switching each other controller based on landing detection algorithm), etc., are the important sub-controllers used in bipedal Humanoid robots. The primary computer will send the reference position data to all motor controllers which then control the DC motors using proportional-differentiation (PD) control (a control feedback mechanism which calculates the difference between the required process variable and the actual value and applies a correction) [7].

2.4 Robot Operating System (ROS)

2.4.1 Different Operating Systems

It is well known that the hardware and software resources of a computer are managed by a set of computer programs known as an operating system (OS). Basic tasks like controlling and allocating memory, prioritizing system requests, controlling input and output devices, facilitating networking, and managing files are performed by the OS. Various services provided by the OS are: Process management, Memory management, Disk and file systems maintenance, Networking capability, Security provision, Device driver's interfaces provision etc. [22].

There are a number of OS, namely, Windows, Mac OS X, UNIX, Solaris BS3000, MS-Dos etc. It is part of a system equipment or computerized equipment which deals with management and coordination of the activities of that computerized system. The computerized system may be a computer, a workstation, a server, a PC, a method, a Smartphone, a road navigation device or any system with some "intelligence" of its own such as a Robot. The OS plays the role of host for all applications running or performed on any hardware.

There are different types of OS [22]:

- Real-time OS
- Multi-user and single-user OS
- Multi-tasking and single-tasking OS
- Distributed OS
- Embedded OS
- Specialized OS like 'Robot OS'

2.4.2 Robot Operating System

It is a set of software structures for robot software development providing OS like usefulness on different category computer system cluster. ROS gives standard OS services such as hardware abstraction, low-level device control, implementation of commonly used functionality, message –passing between processes, and package management. Running arrangements of ROS-based procedures are represented in a graph architecture where processing takes place in nodes that may get, post and multiplex sensor, control, state, planning, actuator and other messages. In spite of the significance of relativity and low latency in robot control, ROS itself is not a Real Time OS; however, it is possible to combine ROS with real time code [23]. The base OS in the present work is Debian in the BBB upon which ROS works.

Software in the ROS Ecosystem can be divided into three groups:

1. Language –and platform- independent tools used for building and distributing ROS- based software
2. ROS client library implementations such as roscpp, rospy, and roslisp
3. Packages containing application-related code which uses one or more ROS client libraries [24].

2.4.2.1 Applications of ROS

ROS may be applied in the following sphere of areas: ROS is a generic term indicating the whole bunch of collection of different heterogeneous software programs (meant for tools, client libraries and application codes contained in ROS) whereas ROS packages include only one group of software's code using one or more client libraries alone of ROS (list at the end of Section 2.4.2).

- A master coordination node
- Publishing or subscribing to data streams, images, stereo, laser, control, actuator, contact
- Multiplexing information
- Node creation and destruction
- Nodes are seamlessly distributed, allowing distributed operation over multi-core, multi-processor, GPUs and clusters
- Logging
- Parameter server
- Test systems

ROS Packages can be gainfully applied in the following areas: One group of software may be applied for say perception, motion etc. while another group of software's may be needed for object identification, stereo vision etc. and so the third group.

- Perception
- Object identification
- Segmentation and recognition
- Face recognition
- Gesture recognition
- Motion tracking
- Egomotion
- Motion understanding
- Structure from motion
- Stereo vision depth perception via two cameras
- Motion
- Mobile robotics
- Control
- Planning
- Grasping [25]

Both the language-independent tools and the main client libraries (C++, Python) are useful for implementing other packages involving functionality and applications such as hardware drivers, robot models, data types, planning, perception, simultaneous localization and mapping, simultaneous tools and other algorithms [25].

2.4.2.2 Benefits of Using ROS

ROS is in actuality a meta-working framework, something between an operating system and middleware for service robotics. It gives not just standard operating services (hardware abstraction, contention management, process management) but in addition system high-level functionalities (asynchronous and synchronous calls, centralized data base, a robot configuration system etc.).

Before robot operating system came into use, each robot researcher and designer would invest impressive measures of time outlining the embedded software within a robot, and in addition the hardware itself. This required skill sets in mechanical engineering and embedded software. Normally, the programs designed in this were very much similar as embedded programming, like electronics, then they were to robotics in the strictest sense, for example, we may experience it these days in service robotics. There was significant re-utilization of programs, as they were firmly connected to the necessary hardware. Another advantage is that of mingling expertise from various fields which incorporates into the designing and programming a robot [27]:

a) Managing the hardware's by writing drivers
b) Managing memory and processes
c) Managing concurrency, parallelism, and data merging
d) Providing abstract reasoning, making great use of artificial intelligence

2.4.2.3 ROS Technology

The following five principles are involved in ROS technology [27]:

- Peer- to peer
- Tools based (microkernel)
- Multi-language
- Thin
- Free and open source

Basic notions in ROS [26]:

- Nodes
- Master
- Topics
- Services
- Bags

Robot operating system is the future of software for autonomous devices. The most critical bit of software for operational robots is the fittingly named Robot Operating System. ROS is a framework of programming tools used to write and develop robot software. It essentially works as a kind of open-source system providing OS like services designed specifically for robotics for example hardware abstraction, device control, implementation of common functionalities and data package management [28]. The rapid development in ROS ecosystem has put it in a path of becoming the Linux of robotic software. The development of robot undertaken in this research study is very gainfully based on and using ROS.

2.5 Requirements of Walking Biped Humanoid Robots

2.5.1 Generic Features

2.5.1.1 Characteristic Features of Humanoid robots

Minimum energy consumption must be aimed at for normal walking of a humanoid, similar to what happens in case of a human body. Achieving dynamic walking is the key to the success of the design and construction of a humanoid. The safety and firm standing on the ground of a moving biped robot is additionally of most prominent concern. Keeping a robot in firm and steady condition by making its center of gravity to remain on or near the midpoint of its weight bearing area is the object of providing capability for its self-control [29]. Set of touch-sensors (tactile) can be utilized to give information on what has been touched. These sensors additionally give data about forces and torques exchanged between the robot and different things [30].

A sensor is a device that calculates a few traits of the surroundings. As one of the most important factor among three essential requisites of a robot apart from planning and control, sensing assumes an imperative part in the ideals of robots, especially that of humanoids.

Sensors are classified i) Proprioceptive sensors and ii) Exteroceptive sensors. The position, the direction and the speed of the humanoid's body and joints are measured by the former while touch (tactile sensors), vision (CCD cameras) and sound (Microphones) are measured by the latter. Actuators are the motors that perform like muscles and joints to the humanoids and mainly rotary actuators are used [30]. They can be electric, pneumatic, hydraulic, piezoelectric or ultrasonic ones.

Planning and control are other two essential requirements of humanoids. Legged locomotion involving biped gait walking of humanoids resemble human-like walking. Ideal planning must aim for minimum energy consumption and effective control means maintaining ideal functions of all parts, particularly, the joints [29].

A robot needs data about contact pressure and its present and future changes in position with a specific end goal to keep up its dynamic balance while it is walking. Another function of humanoid robots is that they move, assemble data (utilizing sensors) about real environmental features and communicate with it. Like manufacturing plant intelligent machines and similar robots that work in exceedingly organized situations the humanoid robots do not remain ideal. Planning and control should concentrate on finding out self-crash locations, path planning and obstacle avoidance for permitting the humanoids to move in complex situations [31].

2.5.1.2 Autonomous Control of Humanoid robots

A robot deciding and functioning on its own, called autonomous robot, is a robot that performs practices and errands with a high level of self-governance, which is especially essential in fields like space investigation, family upkeep, (for example, cleaning, and conveying and providing merchandise and other helps).

Some present day production line robots are "self-governing" inside the strict bounds of their immediate surroundings. It may not be that each level of

opportunity exists in their encompassing surroundings; however, the production line robot's work environment is highly testing and can regularly contain disorganized, unexpected factors.

One critical region of robotic research is to empower the robot to adapt to its surroundings whether this is ashore, submerged, noticeable all around, underground, or in space.

A self-governing robot may likewise study or acquire new information like modifying for new techniques for finishing its undertakings or adjusting to evolving environment.

It is necessary for a humanoid exhibiting human like appearance to perform human like actions independently for it to cooperate and assist people in their everyday life. Interaction among humans takes place through their voice and gestures. Simple gestures arising out of repeated conducts count a lot in developing human communications. Likewise, a humanoid's friendly gestures are important for effectively serving humans. Such gestures should, for avoiding misunderstanding, be in the same way as human does. The way and manner of walking of a humanoid, while approaching a human, is equally important. Dancing or other entertaining services of humanoids must also exhibit meaningful, artistic and creative motions as much possible as human performance [32]. For all this capability the humanoid robot must have autonomous control besides intelligence.

Natural interaction and adaptability to environments meant and designed for humans are the main objectives and aim of the research areas of humanoid robots [32]. Further, robotic technologies that integrate senses, motor responses and intelligence are of great importance. Robot's safety, reliability, and human robot symbiosis calls for autonomous action control for humanoid robots [5]. For fast

dynamic walking, path finding and planning, and avoiding collision an autonomous humanoid robot is only unavoidably and inevitably needed.

2.5.1.3 Intelligence Possessed by Humanoid robots

Autonomous mobile robots are a reality today (not a fiction) on account of the rapid and huge progress in mechanics and electrical engineering fields. Inherent danger and hostile environments (deep-sea explorations, space expedition, nuclear plants, etc.) have welcomed intelligent humanoid robots to work as substitutes for humans. Further intelligent autonomous devices like robotic vacuum cleaners, lawn mowers etc. have flooded the market. Humanoid house keepers, personal assistants will soon take their place.

2.5.2 Essential Functions

2.5.2.1 Walking Control - Literature Review

A core task for humanoid walking robots is the actual walking controller, generally consisting of a gait pattern generator and the balance control. For this problem, there exist two fundamental approaches: one, possibility to set about this matter is to rely on a very accurate model of the walker and to compute gait trajectories trusting in sufficient accordance of the model with reality. The other direction relies on approximating the robot dynamics by a simple model with reduced system states e.g., an inverted pendulum. Accordance of the simplified model with the real dynamics is ensured by feedback control [17].

An often adopted method to make the humanoid robots to walk steadily is to direct them to learn by trial and error and also to physically incorporate the moving positions and their determining factors. Central Pattern Generators (CPGs) techniques enable to produce joint directions, utilizing nonlinear oscillators. In these methodologies, it is a difficult matter to discover accurately applicable determining factors to accomplish a steady walk. Procedures involving much calculation utilize the idea of the Zero Moment Point (ZMP) and are based

on joint positional directions which are figured out considering dynamic movement of the robot. In this case, an exact model of the robot and its movement is required.

A few methodologies have been introduced that go for appropriately limiting properties like speed or mid-body steadiness of a humanoid's walk. The subsequent upgraded walking patterns don't generally look like correct human step. We consider the issue of accomplishing steady human-like stride in a humanoid robot by regarding this as an optimum limiting issue and create four calculation procedures that work on the consideration of joint movement premise [33].

A number of diverse movement controllers and path tracing controls are utilized for human robots. In the ZMP approach, the fundamental target is the determination of robot's movement in a manner that the zero moment point-ZMP (the point where the entire idleness driving pressures add up to zero) does not go beyond the predetermined region of steadiness. Centre of Mass (CoM) based models are another well-known direction-tracing controlling strategy. Every one of these techniques experiences the ill effects of being dependent on planned particulars of the hidden humanoid robot structure.

In the examination work exhibited in this proposition, the emphasis is on a novel direction arranging technique planned particularly for a humanoid robot to move along curved directions. The proposed curved strolling model presented for a humanoid robot utilizes a similar crucial rule adopted in the differential guiding framework for a wheeled robot. A standout amongst the most intriguing standards of this strategy is its independency from the basic humanoid robot structure. This new strategy likewise guarantees a vital effective direction when contrasted with some other outline particular strategies since a few joints e.g., hip move joints are not utilized as a part of this technique to change the direction [34].

Most legged robot endeavors have concentrated either on specific morphology (e.g. biped, quadruped, or hexapod) or a specific motion supposition (e.g. semi static or bouncing). Interestingly, past work by the authors are general, in that they apply free of morphology, and also apply to issues of a class which incorporate legged robots.

Kinematic walking frameworks can be displayed utilizing associations on primary fiber packs and furthermore give outcome controllability. Non-holonomic motion frameworks expect that the conditions of movement for the framework are smooth, which forbids their application to legged robotics issues where the conditions of movement are intermittent. Standard nonlinear controllability tests require that the framework's conditions of movement be smooth. The fundamental commitment of this work is the expansion of standard nonlinear control techniques to a class of issues where the conditions of movement are in broken manner [35].

Up to this point, most steadiness control methods have endeavored to keep up balance by controlling just the straight-line movement of a robot. Strategies have been developed to change the information on joint- degree directions to adjust the position of the Center of Pressure (CoP), a point inside the robot's support region through which the resultant Ground Reaction Force (GRF) acts. At the time when the CoP, resulting from the input joint movement, leaves the safe support base, showing a conceivable toppling of a foot, the movement is altered to bring the CoP back inside the support base while the robot still takes after the craved direct movement of the Center of Mass (CoM). The rotational movement of the robot stays pretty much disregarded in these methodologies.

Be that as it may, rotational progression of a robot assumes a critical part in balance. Investigations on human steadiness control additionally demonstrate that people firmly direct precise force amid walk, which recommends solid

probability that angular momentum, could be critical in humanoid developments [36].

A crucial difficulty with existing systems that create robot's movement is information reliance. For instance, a simple method is to assemble a model for a specific movement from a substantial number of samples. In a perfect situation, the robot could watch one (possibly awful) example of a movement and adopt it to a more human-like partner.

Reliance on huge amount of sample information is frequently an approximate substitute for a more principled approach. For instance, rapidly examining Random Tree [RRT] based strategies offer no authentic human-like movement yet depends upon a database of sample human-like movements posing as if providing a true and correct answer. Verifying the database to discover a movement like RRT-produced direction line is a drawback for web based planning which can affect the procedural calculation period [59].

Other different methods depend upon approximate connections got from the example information to make robot movement to seem more human-like. This takes into account criteria like joint comport, travel time, jolts, human stance to-target connections. At the point when movement information obtained is utilized, often ignoring time details, which causes robot movement to happen at impossible and non-human speeds.

Movement strategies produced for cartoon or virtual characters cannot be appropriately applied to robots since key contrasts exist in the real situations. The degree to which the methods adopted for virtual characters can be used to robots relies on upon the suppositions made for a specific strategy. Limitations like torque and speed cutoff points of real equipment frequently causes movement intended for virtual characters to look poor on a robot, notwithstanding the fact

that those movements look great on a virtual model since fewer limitations exist within the virtual worlds [37].

Motion capture technique [38,39] has been adopted to make humanoid robots move like humans since motion capture is a premium technique for animation of human-like characters in computer graphics.

The stability issues of humanoid robot movement are the critical point in understanding the control techniques; hence this humanoid walking robot can be arranged in three distinct classes. To begin with, the first variety takes into account static walkers, whose movement is moderate so that the framework's security is totally depicted by the ordinary projection of the Center of Gravity, which just relies on upon the joint's position. Second classification includes dynamic walkers, i.e., biped robots with feet and induced ankles. Postural strength of dynamic walkers is provided by joint's speeds and acceleration as well. These walkers are capable of moving in a static way if they have sufficient huge feet and the movement is moderate. The third class involves absolutely dynamic walkers, robots without feet. Dynamic walkers can accomplish quicker strolling speeds, running, stair climbing, execution of progressive flips, notwithstanding that strolling is without any actuators. For this situation the stability polygon in the single-support stage is diminished to a point, with the result that static strolling is impractical. In the stroll with dynamic balance, the anticipated focus of mass is permitted outside of the region enclosed by the feet, and the walker may basically fall in the middle part of the strolling step. The control issues of dynamic strolling are more confused than in strolling with static balance, yet dynamic strolling designs give higher strolling speed and more noteworthy effectiveness, along with more flexible strolling structures [60].

For all the said classifications of strolling robots, the issue of steady state and dependable bipedal walk is the most essential requirement, but then unsolved

with a high level of unwavering quality, this subject has been considered fundamentally through the accompanying two classes of strolling type generators and robot controllers. The initial approach is to produce a powerful steady occasional dynamic strolling design. It is done expecting that the models of robot and environment are accessible, and the kinematic and dynamic parameters of the robot model are accurately characterized. But, the second approach utilizes constrained or disentangled learning of the framework's progression. Be that as it may, for this situation, the control depends much on the input control, and it is important to create strategies without high calculation requirements for ongoing usage [61], [62].

The rotational harmony of the foot is the main consideration of postural insecurity with legged robots. The question has roused the meaning of a few dynamic based criteria for the assessment and control of balance in biped movement. The most well-known criteria are the Center of Pressure (CoP), the zero-moment point (ZMP) and the foot-Rotation indicator (FRI). From these criteria, the ZMP idea has received the increased and broadest acknowledgment and assumed a significant part in settling the biped robot soundness and intermittent strolling design combination. The ZMP is characterized as the point on the ground about which the entirety of the considerable number of snapshots of the dynamic powers works out to zero. In the event that the ZMP is inside the arched structure of all contact focuses between the foot and the ground, the biped robot can walk. Regardless of the possibility that the solidness in light of ZMP just portrays contact condition amongst foot and the ground, ZMP based controller is for the most part utilized as a part of humanoid robot groups since it is known to function admirably. In the vast majority of cases, the direction of humanoid robot is decided off-line, and after that controller is intended to track these directions. A mobile example is created to guarantee that the robot's ZMP is all the time inside the supporting foot projection. This is important for the biped robot to keep up

dynamic balance while walking. Be that as it may, the fancied ZMP of the strolling example is not the same as the real ZMP of the biped robot in a solid walk. Keeping in mind the end goal to make up for the ZMP discrepancies, it is important to execute the balance control utilizing power (F/T) sensor or slope sensor. Most research works managing parity control have concentrated on the pay of ZMP variation on the grounds that the ZMP is the fundamental standard of dynamic balance. By and large, it has been accepted that the principle reason for discrepancy points to the actual ground condition or model errors. As usual, the balance controller ought to be strong against the slope and model imperfections [43].

The two most essential elements of biped humanoid robots are the human-like shape and activities. Biped humanoid robots have two legs and should stroll with a decent portable capacity on different territories including uneven surfaces or stairs. Accordingly, numerous analysts have created humanoid robot structures and have contemplated the biped strolling of humanoid robots. The Honda humanoid, the WABIAN arrangement of Waseda University, H6 and H7 of Tokyo University, HRP of AIST and JOHNNIE are notable human-scale biped humanoid robots. For the most part, the control system of dynamic strolling of biped robots depends on the walking pattern generation, which considers the steady zero-minute point (ZMP) direction and online balance control. As the genuine ZMP direction is not the same as the required ZMP direction because of reasons like the unevenness of the surface, detecting blunders and blemished dynamic model of the robot, a few online controllers in light of the tangible criticism are required. Also, numerous other researches consider the stabilization control strategy of humanoid robots identified with the angular momentum information.

A dynamic strolling control technique has been proposed by Kim *et. al.,* in 2006 for biped humanoid robots utilizing the ZMP and inertial data. The control

procedure covers adaptive walking pattern generation, actual ZMP adjustment of error while standing on single leg with damping control of the ankle joint, stable landing control and landing position control depending on the precise speed of the upper body. In this way, a biped robot can adjust to uneven landscape without losing steadiness at the time of actual strolling [44].

The two most basic components of biped humanoid robots are the human-like shape and advancements. Biped humanoid robots have two legs and ought to walk around a tolerable compact limit on various domains including uneven surfaces or stairs. In like manner, various examiners have made humanoid robot arranges and have thought about the biped walking around humanoid robots. The Honda humanoid, the WABIAN game plan of Waseda University, H6 and H7 of Tokyo University, HRP of AIST and JOHNNIE are eminent human-scale biped humanoid robots. Generally, the control arrangement of component walking around biped robots relies on upon the walking outline period, which considers the relentless zero moment point (ZMP) course and online conform control. As the honest to goodness ZMP course is not the same as the looked for ZMP heading as a result of reasons, for instance, the unevenness of the surface, recognizing bungles and imperfect component model of the robot, a couple of online controllers in light of the substantial feedback are required. Additionally, various other research wears down the conformity control arrangement of humanoid robots related to the exact drive information have been disseminated. In any case, there have been decently few research manages ZMP control and alter control using inertial estimation at the same time [45].

In this thesis, a dynamic walking control procedure is proposed for biped humanoid robots using the ZMP and inertial information. The control scheme fuses flexible walking plan period, progressing ZMP compensation in the single support stage with damping control of the lower leg joint, stable landing control and landing position control in perspective of the exact speed of the center. Along

these lines, a biped robot can change in accordance with uneven scene without losing strength logically in the midst of walking.

There are three kinds of methodologies in the case of biped strolling, which are offline pattern generation, offline pattern generation with online feedback compensation, and online pattern generation with online feedback control. The first is to create the strolling designs in the wake of outlining ZMP directions in light of the exact information of the framework, for example, the snapshot of dormancy, masses of every part. This approach can get the steady strolling designs, yet it can make the robot to tumble down effortlessly, in light of the fact that it is touchy to the un-displayed or obscure elements of the framework like response and erosion strengths with the ground. The strolling design itself is steady, yet we ought to ascertain confused ZMP elements to get the strolling design. The second one is to defeat the security and the strength issues. This approach utilizes ZMP based strolling designs and repays them to keep adjust by criticism. It can build, utilizing input controller, the strolling dependability of the robot by diminishing the unsteadiness elements prompted by displayed flow, ground conditions and so forth, yet the unpredictability of stable strolling design era still remains [45].

The third approach is utilized as a part of this work. The online walking pattern is created by the kinematical approach by producing the positional dictates of the joint. It is made by watching the human's conduct, and changed by tactile input controllers to keep the strolling solidness. This implies strolling designs have been separated into kinematic reference generation and dynamic controller. The strolling design is redesigned such that it can roll out the robot improvement, its progression time and walk without halting. The position bends of the pelvis focus are figured as for the limit conditions (position and speed) toward the beginning and toward the end of the progression, since they utilize third order polynomial addition. We can correct the form of the curve by selecting the correct limit

condition values by considering the strolling modes (forward, backward and side walk), recurrence and walk, which can be the directions from the administrator or from the route directing calculations [45].

The feedback controllers used in this work are named to ZMP, landing orientation, landing position, landing timing, damping and vibration reduction controller on the point of their objectives and functions They utilized the F/T sensor on the lower leg and the accelerometer on the sole, and they were very much executed on KHR-1, KHR-2, KHR-3(HUBO) to keep the robot's security in the altered stride time and walk strolling condition. We utilized the controller exchanging strategy regarding the planning, and the period of the walk. This implies we are applying the diverse controllers with various circumstances. This technique can be a decent approach in the state of altered stride time and walk. We extended this way to deal with the variable stride time and walk condition amid strolling in this work [45].

Early biped strolling of robots included static strolling with a low strolling speed. The progression time was more than 10 seconds for each progression and the balance control system was performed using CoG (Center of Gravity). Thus, the anticipated point of CoG onto the ground at all times falls inside the supporting polygon that is made by two feet. While static strolling, the robot can stop the strolling movement at whatever time without tumbling down. The weakness of static strolling is that the movement is too moderate and wide to shift the CoG.

Most biped humanoid robots have performed stable dynamic strolling on the effectively got ready level floors. Strolling studies on the uneven and sloping floors are still in the early stage. Dynamic strolling on an uneven surface is difficult to be obtained in light of the fact that most biped humanoid robots perform hard position control of the joints by utilizing motors and lessening gears and the reaction times of the actuators and sensors are low because of the gear

and sensor sound. So then, it is unimaginable for the robot to gauge the ground conditions promptly and it is likewise unthinkable for the robot to properly react regardless of the possibility that it quantifies the ground conditions quickly. But, the human ankle can quickly adjust to varying nature of the field. Besides, human muscles can contract or unwind rapidly with smooth movements [45].

This work by Kim *et. al.,* in 2007, depicted a dynamic strolling control calculation procedure that considers nearby and far-off slanting state of the floor. The writers propose the utilization of different online controllers to adapt to an uneven and sloping floor in the light of an upgraded form of a formerly proposed dynamic strolling calculation. These online controllers are initiated and made to work one after another during appropriate time in a mobile cycle [46].

2.6 Conclusion

The following observation and analysis of bipedal humanoid walking is taken into consideration in this research study:

There are different reasons for difficulties in control issues to arise and different works and stipulations that must be reasonably settled and satisfied keeping in mind the end goal of making substantial strolling and other performance actions of humanoid robots. Past investigations of organic nature, hypothetical and PC outputs have concentrated on the structure and choice of control calculations as indicated by various criteria, for example, lower power utilization, making energy available at all times, strength, speed, solace, capacity to keep moving, and environment affect. All things considered, notwithstanding these perspectives, it is likewise important to consider some different issues: capacity of mechanical execution because of the physical constraints of joint actuators, adapting to complex and strong nonlinear progression and instabilities in the model-based approach, complex nature of repeated and regular walk, incorporation of learning and adjustment abilities, calculation issues, and so forth.

The significant issues connected with the examination and control of bipedal frameworks is the highly-coupled nonlinear dynamics and in addition, the discrete changes in the dynamic phenomena because of the way of the stride. Regardless of the humanoid robot structure and many-faceted quality, the fundamental qualities for every bipedal framework are: a) the DOF shape developed between the foot and the ground is one-sided and under incited b) the repetition of walk (symmetry) and consistent compatibility of the quantity of legs that are at the same time in contact with the ground. while walking, two unique circumstances emerge one by one: the statically stable two-leg standing stage in which the robot is carried on both feet at the same time, and statically weak single-leg standing stage when just a single foot of the instrument is in contact with the ground, Thus, the moving method changes its structure in a solitary strolling cycle from an open to a shut kinematic chain [43].

3.Integrating Experimental Robots

3.1 Advanced Multifunctional Quadruped BeagleBone Black Robot

Robots have a tendency to do a few or entire functions shown herein: receipt of electronic programming, handle information or physical recognitions electronically, work independently to some extent, roam around, carry out its own physical parts or physical procedures, realize and control their surroundings, and display smart activities [28,50]. The word robot can allude to both physical robots and virtual programming operators, but it is usual rather to call the second ones as bots [51]. They can be remotely controlled, semi-autonomous or autonomous.

Internet is available everywhere and likewise information technology has come to every household and so new intelligence is growing among people. Robot technologies also, alike the human, integrate senses, motor responses and therefore must possess intelligence. In addition, devices meant for constructing robots are also rapidly improving. This calls for new research study in industries and universities [5].

3.1.1 Developments in Robotic Research

Servo control and trajectory planning are important research topics. Space, undersea, and other hazardous habitats (nuclear power plants) unavoidably needed robots and hence adaptability became the focal point of research, of which the intelligent robot represents a typical class. When harsh workplaces like underwater, nuclear power plants, distance unknown space etc. invited robots with open hands, knowledge and understanding of the physical environment and methods for navigating such terrains turned out to be key research issues. Quadruped walking robotic machines were preferred to biped humanoid walking robots where flexible walking motion using four legs became a necessity for robots to move rapidly through environments having lot of steps, piping, and other obstacles met with in extreme working conditions such as nuclear power

plants. Smooth contact of feet with ground, lowering the peak torque, and reducing energy consumption were the difficult problems faced. Animal style walking was copied for robots with four legs.

3.2 Advanced Biped Bioloid Humanoid Robot

3.2.1 BeagleBone Black as 'Deciding and Enforcing Component'

BeagleBone Black (BBB) is a commercial 'deciding and enforcing component' which is mounted on a four legged vehicle with Lynx motion and driven by it (cf. Figure 5). It is equipped with several sensors such as sonar sensor, Web scam, GPS, etc. Such a versatile intelligence providing component has, however, not been endowed with the provision of mounting on and operating a biped humanoid dynamic walking robot and this deficiency is cured in this research work. Thereafter, this research study takes advantage of the powerful computing capability of this fortified BBB and mounts it on the structure of a commercially available humanoid robot (after removing the latter's own less-efficient deciding and enforcing component) through the ROS technology and Python language based software. This is a first of its kind research in this context.

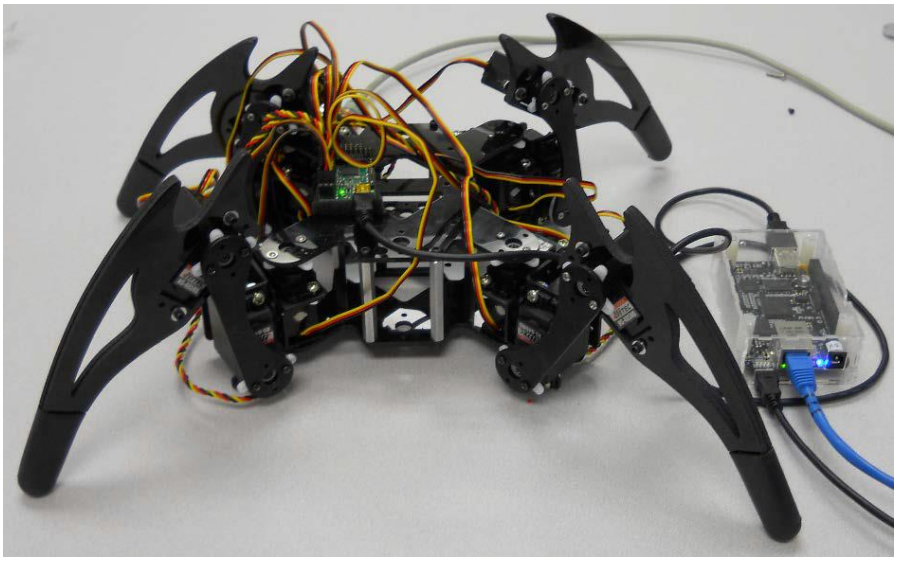

Figure 5: BeagleBone Black Mounted on a Four-Legged Vehicle

Source [52]

3.2.2 Features of Bioloid Premium Humanoid Robot (BPHR)

The BPHR manufactured by the Korean firm Robotis stands tall among them especially with its structural components. In the humanoid type there are three Bioloid robots namely, Premium, Grand Prix (GP) and Darwin models.

The Bioloid commercial kit is convenient, safe and also expandable. Generally, they are meant for building a robot including a humanoid for education, entertainment etc. purposes. BPHR includes gyro/accelerometer, DMs, and multi-channel wireless expandable remote controller. These humanoid robots have been designed to be completely modular. Semi-transparent humanoid skin is used.

As a remote control humanoid robot BPHR can walk more safely. But it lacks autonomous operation, if autonomous operation is desired; different controller has to be used. The product contains an extensive motion library. Accelerometer system is found to be active during walking and so the coefficients need tweaking. Therefore, software is needed to be included so that it could be turned off and on. Similarly, it is found that if complicated sensors such as a compass or vision system are to be included then a different controller had to be used. The controller provided in BPHR is not designed for a lot of decision making and for including many sensors. An autonomous humanoid robot requires more than one processor or at least one that does multiple threads. Many users have evaluated the BPHR to be good in its class of limited use. Wikipedia, the free encyclopedia, in a write up on Robotics Bioloid says that the Bioloid framework is in this way similar to the LEGO Mind storms and VEXplorer sets. Hence the structural component of Bioloid humanoid robot is chosen for this study. This enables the autonomous capabilities of the robot with the help of feedback from the accelerometer and the IR sensor. The addition of the wi-fi capability further helps in the enhancement of the autonomous activity making the robot to be remotely controlled.

3.2.2.1 Structural Components of BPHR

BPHR (Bioloid Premium Humanoid Robot) has a light-weight aluminum frame that comes with eighteen Dynamixel servo motors (eight AX-12 motors for the chest-to abdomen region and ten AX-18 motors for the waist and below area), which are integrated motors with several features, such as position. Speed can be controlled easily with feedback for angular position and angular velocity etc. Load torque can be set up in the motion. There are about 255 motions and each motion can be further manipulated by adding steps in the software to any of the specific motions made by the user [53]. The Bioloid humanoid robots are good walking ones as they self-adjust posture while walking. Wiring is with daisy chain connection and supports many Dynamixel units with very few resources. Three types of assembly construction are provided. In addition, other types of assembly are supported. The limitation however is the actuators in the level field do not support enough torque and therefore some of the pose cannot be completed in a type of assembly [53].

3.2.2.2 Enhancement of BPHR's Capabilities

Since 2000, there have been many challenges in developing robots that can safely share spaces with ordinary people and provide helping and taking care services. The main focus has been on human- robot interaction and autonomous mobility [5]. Therefore, there is need for making the BPHR to have autonomous control. Humanoid robots must also be service oriented for practical use. Therefore, they must be capable of switching between different modes of travel, viz. on the surface of water, under water, air and distant space. The decision making and enforcing component must provide to the Bioloid humanoid robots all these capabilities. Voice control is also an important consideration for humanoid robots and so microphones must be mounted for remote speech recognition.

3.2.3 Controller of BPHR

3.2.3.1 Processor

There is no computer brain for the BPHR as it is only an educational and entertainment one. A humanoid robot meant for assisting and cooperating with human beings must possess intelligence and for this purpose at least a single computer or notepad is essential. The brain in the form of processor available in BPHR is only a microcontroller. BPHR has CM-530 controller only, which has the following specifications:

- CPU: ATMEGA 2561
- Internal I/O device, 6 buttons, Mic, Temperature sensor and voltage sensor
- I/O device: I/O & 6 of 5P I/O for analog sensor installation

This microcontroller has 64 pins which control individually a part of the robot, such as the part determining input and output; and amount of voltage used by the robot. This may have about 1KB RAM and 15KB memory against 4GB RAM and 1000GB memory of a computer. This is quite inadequate for human-like walking and other actions.

3.2.3.2 Controllers

Cerebellum of the brain (controls and correlates the motions of actuators) is the CM-530 controller carrying inside it an ARM Cortex STM32F103RE microcontroller. This is a fairly powerful one. But it has to be changed as it is not having multifunctional capability except for biped humanoid walking ability.

3.2.3.3 Software

Operating system (tools and library for conveying and collection of routine information) is having improved software by using embedded C instead of the RoboPlus, because every part of the robot is programmed individually by the programmer. It is capable of multi-threading also. It encapsulates three

components, namely, Motion file, Task file, and Manage file. These three files implement the robot's functionalities, configure the programming and set up each actuator and other accessories in a user friendly interface. Further, these components separate the hardware configuration and parameters from software programming mechanisms. But the communication of BBB is with the RC remote controller only.

Thus the overall position of the intelligence providing component of the Bioloid robot is insufficient and hence it is proposed in this research work to replace it with the BBB intelligence providing component.

3.3 Replacing the Deciding and Enforcing Components

The following changes have been necessitated and made to develop the humanoid robot in this study in order to replace the less-efficient deciding and enforcing (intelligence providing) components of BPHR with that of BBB (after strengthening and enabling the latter to possess bipedal humanoid walking capabilities also in addition to its multifunctional capacity).

The above changes are shown graphically in Figure 6.

Table 1: Bioloid Robot listing existing and replacement components

Sr. No.	Particulars	Existing Components of Bioloid, GB Robot	Replacing Components from BBB
1	CPU	STM32F103	AM33588
2	Controller	CM530	Beagle Bone Black
3	Remote Controller	RC100	Laptop Interface
4	Software	RoboPlus with C++	ROS with Python
5	Wi-Fi	Zigbee	Wi-Fi Adapter
6	Camera		Web Camera

Figure 6: Schematic Parts of Humanoid Robot

Source [54]

A collection of nodes and programs called roscore are the pre-requisites of any ROS-based system. There must be a roscore running in order for ROS nodes to communicate [63]. Communication buses named ROS Topics are used while using the ROS modules. These comprise of anonymous publish/subscribe semantics that in-turn decouple the production of information from its consumption. Generally, the nodes are unaware of the extremities with which they are communicating. The nodes that are in search of data subscribe to the relevant topics i.e. the nodes that generate data publish to relevant topics. Thus, there can be multiple subscribers and publishers to a topic [64]. An rqt_graph is commonly used as a GUI plugin for visualizing the ROS computation graph [65].

4. Experimental Procedure and Results

This chapter discusses the adaption strategy for improving the capability of the Robotis make Bioloid humanoid robot by substituting its built-in CM530 controller with that of BeagleBone Black (BBB). This also describes the actual walking performance of the robot with the final results obtained.

4.1 Adaption Strategy

The default CM530 controller within the Bioloid robot is replaced with the BBB controller. Further, the BBB's multi-functional deciding and enforcing component capabilities is improved to human-like dynamic walking ability which was not possessed by it earlier.

The Robotis make biped BPHR is composed of the following components:

i. Dynamixel Motor (AX-12A)
ii. Gyro/accelerometer sensor (MPU 6050)
iii. Infrared sensor (IR)
iv. CM530 controller

Generally, a walking humanoid robot requires to be configured with an autonomous controller and a main controller. The autonomous controller takes care of the autonomous processing and any associated image processing tasks. Both a Computer and a microcontroller may be used as the above controller or use can be made of just a microcontroller all alone relying upon the performance needs of the robot. The main controller controls the motion controller having programs for walking and further controls a special action program (meant for guiding functions of other actuators) written in a suitable programming language, like C, C++, and Python.

Keeping these basic and fundamental design requirements of a humanoid robot the adapting strategy is evolved. Deciding cum enforcing components of BPHR

are replaced under this strategy with BBB's such components after enhancing it to perform biped humanoid walking which was not there earlier. This enhancement was made possible using ROS and Python. In BPHR, the CM530 controller is the key for governing all the robotic movements and the control of the motors. The idea is to replace this default controller with a different controller which is easily programmable and can make the working of the robot more efficient. The deciding cum enforcing intelligence providing components of BBB Board (incorporating Robot Operating System (ROS) and Python libraries and walking control) are implanted in the BPHR. While carrying out this process the original decision making components are removed from the BPHR. This new method has been implemented in this thesis study. Such an installation procedure has been undertaken for the first time in this thesis work as far as the researchers' knowledge goes.

4.2 Assembling the revised Humanoid Robot

BBB is a powerful processor-cum-intelligence providing component that can be used in a humanoid robot Structure with required alterations made herein. The structural platform of the BPHR is constructed with 2 legs to provide 12 degrees of freedom (DOFs) using 12 Dynamixel servos that are connected to it. All the Dynamixel motors are linked in daisy chain pattern. One side of the first servo is linked with the 12V power source and the other side of the servo is linked with the USB2 Dynamixel connector. The complete interface of the BeagleBone with the robot is shown in Figure 7.

The two-legged robot is constructed as stated, using 12 servos resulting in 12 Degrees of Freedom (DoF). USB2Dynamixel connector is used to operate the Dynamixel actuators through the Wi-Fi interface of BBB. Position of the Dynamixel AX-12A servos is obtained using inbuilt encoders. A Gyro/accelerometer sensor is mounted on the robot, which supports in the

balancing the robot. This accelerometer sensor updates its parameters periodically [56]. Infrared (IR) sensor fitted on the robot's upper body is utilized to identify objects in front of the robot that prevent it to move on. Additional libraries are added from ROS, thereby enabling the BBB to work with a Wi-Fi adaptor.

All the Dynamixel servo motors are controlled by using the python program. The accelerometer sensor fixed in the body of the robot help to detect the stability of the robot. Thus it controls its balance for walking so that it does not topple down. A wireless network is used to provide communication between the computer and the robot.

4.3 Humanoid Robot's Dynamic Walking Algorithm

A detailed consideration of many different walking methods and their drawbacks as well as their analyses is made in chapter 2. Based on this, the algorithm, Trajectory, and Walking Pattern Generation method adopted in 2006 and 2007 was most appealing and found highly suitable and desirable and hence in this

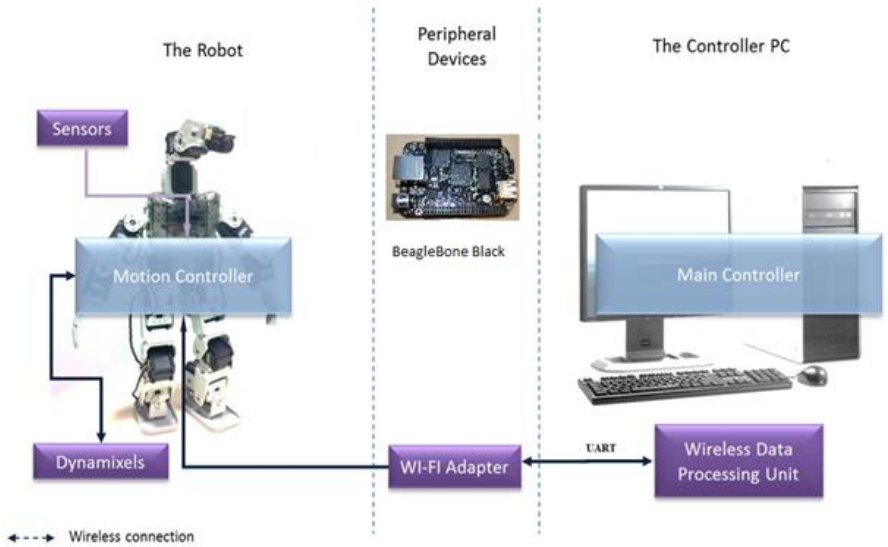

Figure 7: BeagleBone Black Robot's Component

Source [55]

study, the walking control strategy developed in references [45,46] are adopted and discussed in detail in this section.

4.3.1 Evolving Dynamic Stable Humanoid Robot Walking

A Debian image with ROS indigo implementation is installed on the Beagle Bone Black. A Wi-Fi module is enabled in the BBB using the necessary libraries and repositories which helps in maneuvering the robot freely. A Dynamixel2USB connector is responsible for enabling the dynamic actuators is connected through Wi-Fi interface of the BBB. Accelerometer sensors are connected to the Serial Clock (SCA) and the Serial Data (SDA) pins while the Infrared (IR) sensors are connected to the analog to digital converter (ADC) pins of the BBB. The IR sensor senses any obstacle in front of the robot and if there is any obstacle in front then it will stop the further movement of the robot (cf. Figure 8). Dynamixel servos use serial communication.

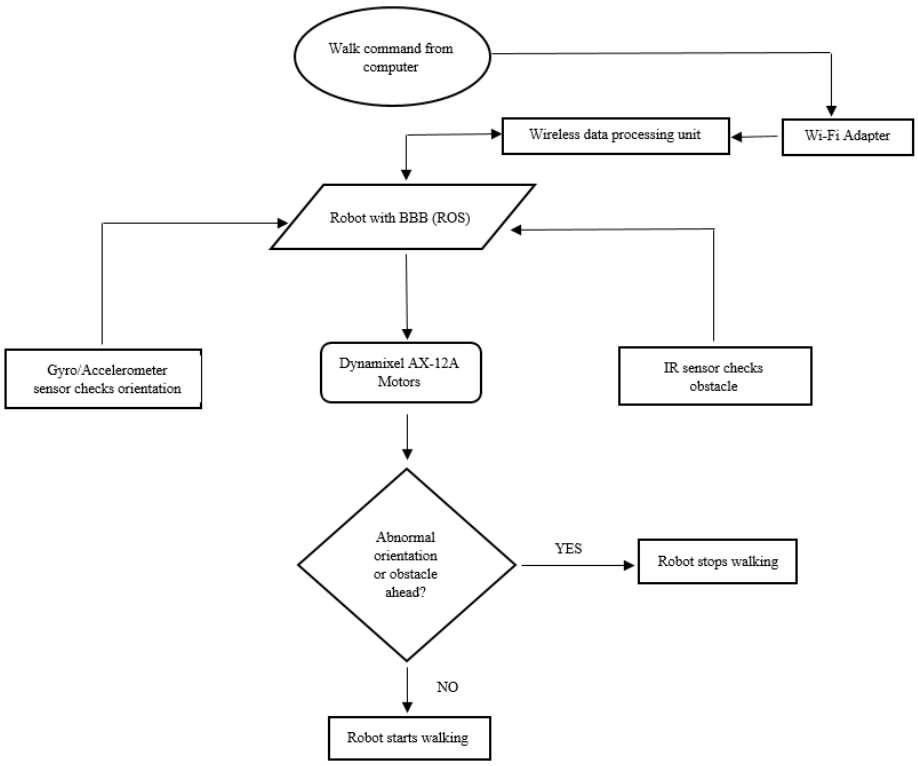

Figure 8: Flowchart Indicating Control of Robot Movement

The default controller baud rate is 1megabits per second (Mbps). ROS acts as the communication intermediate between the computer and the BeagleBone Black. ROSCORE is the first thing we should run in one terminal when using ROS. ROSRUN[13] & ROSNODE[14] should run in another two new terminals.

The walking pattern of the robot is fine-tuned so the robot moves in small steps while balancing itself. The location of the pelvis center and ankle as seen from the sagittal plane is exhibited in Figure 9 & Figure 10.

When the robot is not in motion, both the feet lies on the ground firmly. However, at the time when the robot begins to walk, the robot is made to swing to the right

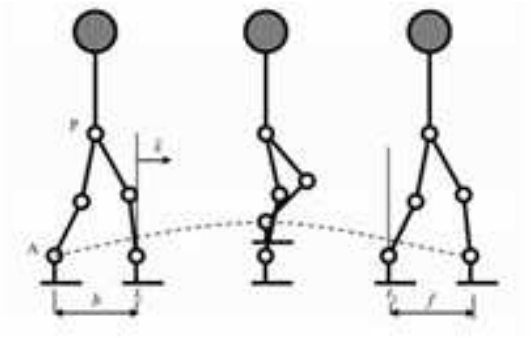

Figure 9: Sagittal View for Walking Pattern

Source [45]

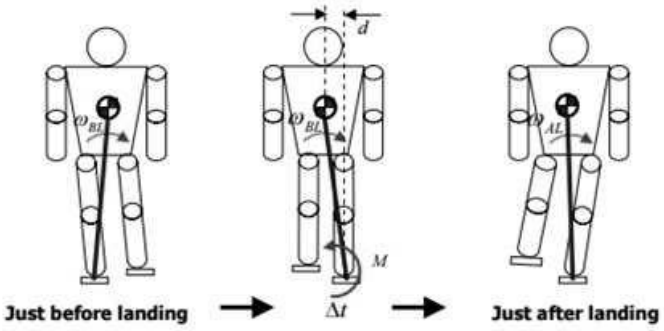

Figure 10: Schematics of Landing Position Control

Source [46]

[13] rosrun is a keyword which enables to directly run a node within a package in ROS.
[14] rosnode displays information about the ROS nodes.

side so that the left leg is made to lift and subsequently moved forward and placed back on the ground. During this time the right leg is tilted to the right side to make the center of gravity shift and this in turn prevents the robot from falling. The process is repeated when the robot is made to tilt to the left side for being stable and keep up its posture so that the right leg is lifted and moving forward and in this manner the robot moves ahead. During these robot movements, the accelerometer sensor plays an active role by sensing the swing and movement through its X and Y axis. The accelerometer sensor continuously sends data to the BBB and the computer so that the sensor parameters are updated periodically.

4.3.2 Stability Analysis

This section details the heuristic approach undertaken to set the parameter values for individual sensors so as to prevent the robot from falling. When the parameters go beyond the prescribed limits inscribed in the accelerometer sensor datasheet then the robot is made to swing to the other side to counter the balance and the robot is thus prevented from falling. When the robot moves forward, if the IR sensors detect the presence of an obstacle, then further movement of the robot is stopped. This dynamic model is utilized as a building block to propel the servos mounted on the leg, thereby resulting in a swing-stance period of the legs for further movement [57]. These parameters have been obtained using a heuristic approach and then these values are set as limiting factors for the sensors in the software.

The RoboPlus software of the BPHR is a symbol form C-language based software meant for simple programming and managing motion and behaviors. The present work replaces the RoboPlus software with ROS and python scripts for flexible control of the autonomous robot. The complete code used for controlling the robot in the present work is detailed in APPENDIX A.

4.4 Design and connections of components for the desired walking pattern

4.4.1 Configuration of BBB

The BBB comes pre-installed with a Debian image which needs to be upgraded prior to the configuration. A fresh installation of the Debian image is also suggested as the pre-installed image has stability issues. A Debian 7.5 version is installed on the eMMC (embedded Multi-Media Controller). This version is stable and is configurable to the requirement of the task (cf. Figure 11 and Figure 12).

4.4.2 Dynamixel AX-12A servo actuators

The dynamixel AX-12A servo actuators are used as they are considered to be the most advanced actuators available. These have the ability to track its speed, voltage, temperature, shaft position and load. All individual servos can be accessed independently to control their speed and their positions. The position and sensors are handled by the servo's built-in the microcontrollers (cf. Figure 12).

Figure 11: Figure showing the BBB, AX-12A motors, USB2Dynamixel connector, Wi-Fi Adapter, IR Sensor and the Gyro+Accelerometer sensor used in the experiment.

4.4.3 USB2dynamixel connector

This is an indispensable device that is required to connect the dynamixel motors to the PC. It connects to a USB port of the PC to the dynamixel motors via its 3p connection port. It uses the Transistor–transistor logic (TTL)[15] network for this work (cf. Figure 11 and Figure 12).

4.4.4 Infrared Sensor

Infrared (IR) sensor GP2Y0A41SK0F is used in this work for obstacle detection. This is achieved by the transmission and reception of an infrared signal. This signal is emitted by the signal emitter on the sensor which traverses back to the receiver after being bounced by the obstacle's surface (cf. Figure 12).

4.4.5 Accelerometer

The accelerometer used here is the MPU-6050 Accelerometer + Gyro. It determines the acceleration. Although a three-axis accelerometer could identify the orientation/tilt of a platform (here this is done for the robot) relatives to the earth's surface (cf. Figure 12).

4.4.6 Wi-Fi-Adapter

A USB Wi-Fi adapter (Logic) with a Wi-Fi hub Advent (HB212) is used for the wireless connection for communicating with the BBB (cf. Figure 12).

4.5 Walking procedure of the robot

There are a total of 12 dynamixel-AX12A servomotors used to make the robot achieve the desired walking ability. The motors are numbered from M7 to M18. To provide the movement ability at the ankle position of the robot motors

[15] Transistor–transistor logic (TTL) is a class of digital circuits built from bipolar junction transistors (BJTs) and resistors.

M15 and M17 assist the right leg and the motors M16 and M18 assist the left leg movement. M13 and M14 serve as the knee movement motors for right and left

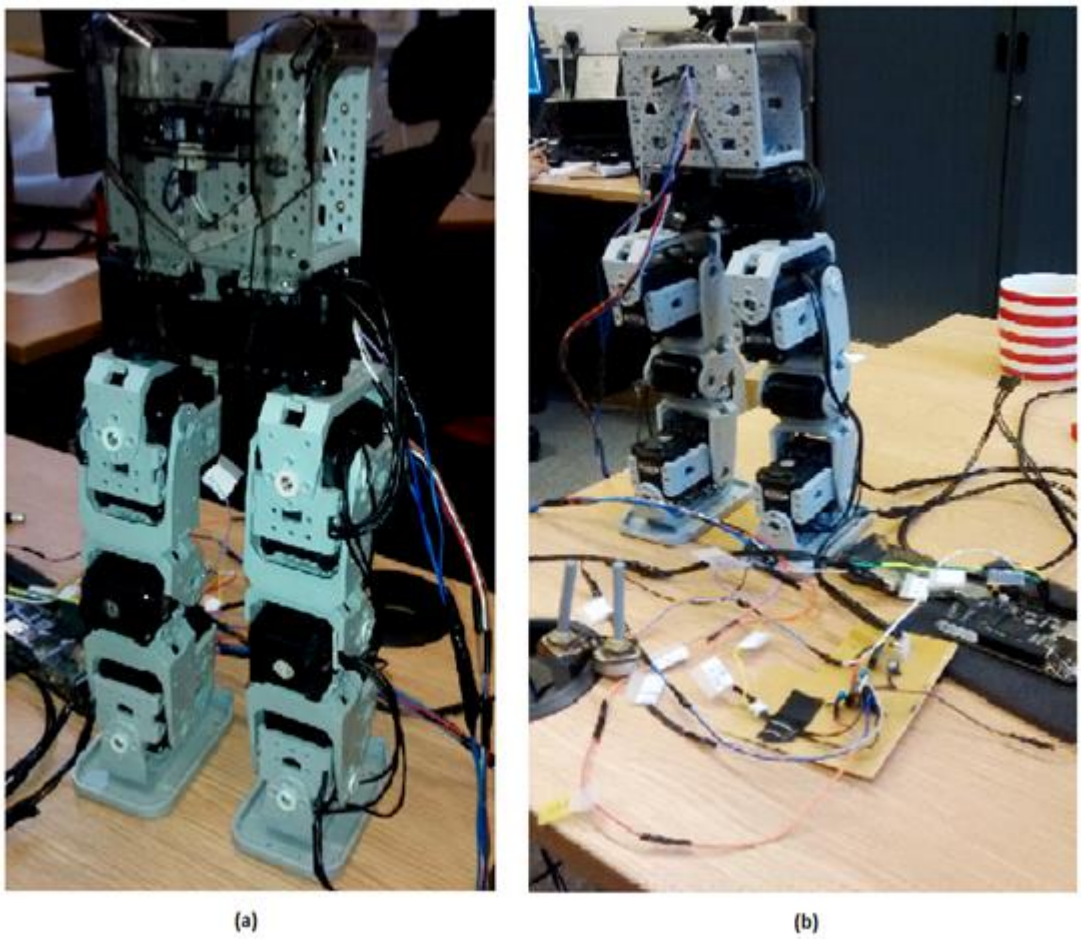

Figure 12: Actual motor positions showing (a) Posterior view (b) Anterior view.

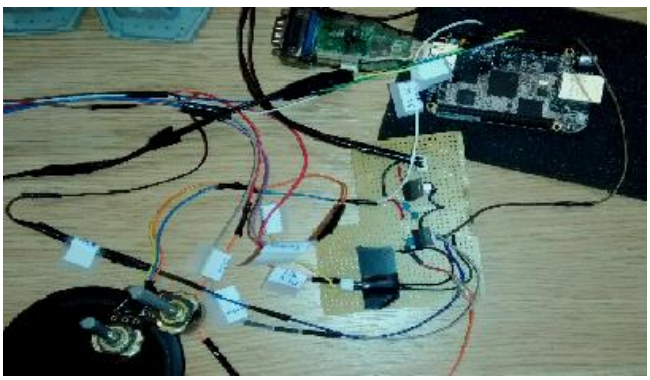

Figure 13: Physical connections of the USB2Dynamixel with the BBB

knee respectively. There are three motors each on every hip position assisting in the resulting movement. These are M7, M9 and M11 on the right and M8, M10 and M12 on the left of the robot (cf. Figure 13 and Figure 14).

The motors M7 and M8 on the hip position are the YAW motors. M9, M10, M17 and M18 are the PITCH motors where, M9 and M10 are on hip and M17 and M18 are on the ankle. M11, M12, M13, M14, M15 and M16 are ROLL motors. M11 and M12 are at the hip position, M13 and M14 are the knee motors and M15 and M16 are at the ankles (cf. Figure 13 and Figure 14)

4.6 Walking strategy of the robot

When the initial command ROSRUN for the movement of the robot is given then there are two options for the speed of the walking of the robot i.e. slow and

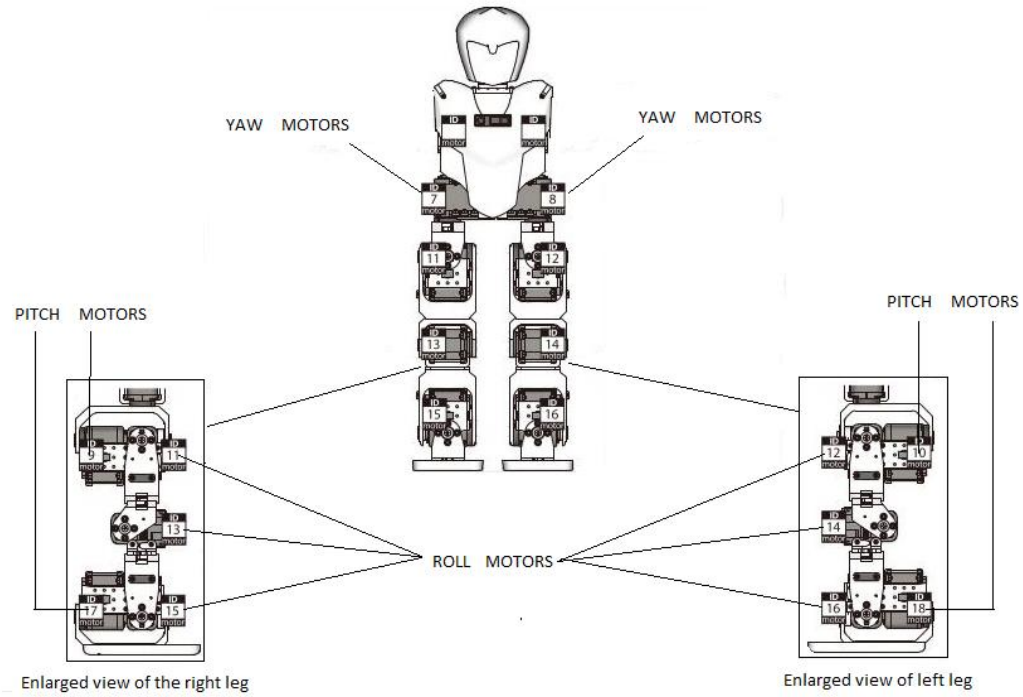

Figure 14: Schematic Parts of Humanoid Robot showing the motor positions for Yaw, Pitch and Roll motors.

normal. It is required to provide an input as 1 (slow) or 2 (normal) for further movement of the robot.

In the very first step, all the motors associated with the walking movement of the robot are set to the default resting position (refer APPENDIX A). This is the standstill position of the robot. All the four pitch motors come into action and make the robot lean on its left side. The angle to which the robot leans is governed by the feedback from the accelerometer sensor. This angle is 20 degrees. If the robot tilts beyond 25 degrees (margin considering the inaccuracy of the sensor in determining precise degree), then it would come back to its standstill position and stops further movement. After it leans successfully to 20 degrees, another motor M10 comes into action. This M10 motor gets back to the initial default position which enables the robot to lift the right leg. Then, the motor M11 swings it forward and the motor M13 brings the knee down. The IR sensor continuously checks for the obstacle in front. If an obstacle is detected around 35 cm, then the robot stops further movement and comes to its standstill position.

In the second step, all the pitch motors make the robot lean on the right side and the motor M9 comes to its initial default position thereby enabling the robot to lift its left leg. Motor M13 is made to come to its default position so that the whole of the robot comes forward. Motor M12 swings the leg forward and the motor M14 brings the knee down. This marks the completion of step two.

In the third step, all the pitch motors make the robot lean on the left side and the motor M10 comes to its initial default position enabling the robot to lift its right leg. Motor M14 is made to come to its default position so that the whole of the robot comes forward. Motor M11 swings the leg forward and motor M13 brings the leg down, completing the step three.

Steps two and three are repeated in succession so that the robot continues its bipedal forward movement. A check is also made continually to see that the robot

is in a stable position. This makes it to avoid falling, if the parameters are more than the prescribed limits inscribed in the sensor. The angle to which the robot leans is governed by the feedback from the accelerometer sensor, which is 20 degrees. If the robot tilts beyond 25 degrees, then it would come back to its standstill position and stops further movement. When the robot moves forward, the IR sensors sense obstacles in front of the robot. If an obstacle is detected to be present at around 35 cm, then the robot stops moving further and comes to its standstill position. This dynamic model can be used, as a building block, to actuate the motors on legs and joints for a swing-stance period of the legs for its further movement. The maximum walking speed of the robot is 0.5 feet/second. Beyond this limit the robot becomes unstable and falls down. Slippery or uneven surfaces also affects the robot and it stability. The rqt graph for the connections during the walking of the robot is shown in Figure 15.

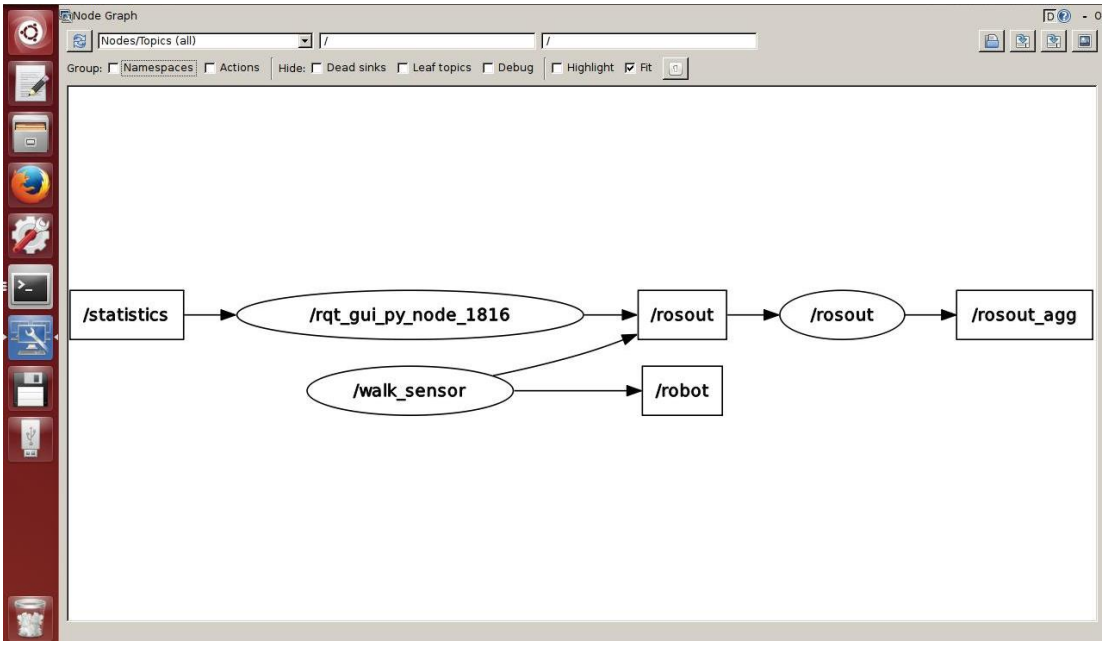

Figure 15: The rqt graph of the connections during the walking of the robot.

5. Conclusion and Future Directions

One of the inspirations driving the operational range of humanoid robotics is to create robots that are better suited to situations intended for people and that these robots are fit for informal intermingling with people [33]. Robotis make Bioloid Premium Humanoid Robot (BPHR) is a humanoid bipedal walking robot capable of simple functions that mimic humans. This robot's deciding and enforcing components providing intelligence can only perform, as stated, simple functions at homes, hospitals etc. This intelligence providing component is removed, retaining only its structural components, namely, manipulator, end-effector and actuators. In this structural body, the multifunctional deciding and enforcing, i.e. intelligence providing component of BeagleBone Black (BBB) is imbibed. Thus, this humanoid robot possesses both the bipedal dynamic stable walking capacity as well as the advanced multiple functions of robotic machines like BBB.

The two-legged dynamic and stable walking capability is incorporated into the BBB's by developing a walking pattern with obstacle avoidance capacity algorithmic code through the usage of ROS and Python Language.

Thus, in this thesis four very important research achievements have been obtained. The first is, developing a method to combine the two different components of two readily available robots in the market. The manipulator, end-effector and actuators of BPHR were selected as a standard robotic structure. This robot's deciding and enforcing component was removed as it had only biped walking capacity and simple (without other advanced) human-like functions. Within this structure the highly advanced and multifunctional deciding and enforcing component of BBB was installed after vitalizing it with human-like walking capacity using ROS Software with Python Script and libraries. BBB did not possess bipedal human-like walking power previously. Thus, this new

humanoid robot possesses all functions that are desired in a latest humanoid class robot. Bipedal walking capability has been acquired by humans over time making human stand ahead among the various life forms. These capabilities when enacted by the robots will make them more human-like and will provide an upper position to others in comparison to the other locomotor robots with time.

The second significant achievement is the biped walking ability that was imbibed to this new hybrid humanoid robot using ROS and Python libraries and walking algorithmic codes and controls for the first time in this research study.

The third important result of this research study is making the hybrid humanoid robot developed in this study is made to possess autonomous control of its walking behavior. For this purpose, the hybrid humanoid is Wi-Fi enabled incorporating suitable and necessary libraries. The robot is capable of performing a bipedal movement without falling down and also getting the feedback from the IR sensor for and obstacle in front.

The fourth important achievement is that this hybrid humanoid robot is made capable of avoiding obstacles by installing IR sensor i.e. the robot stops when it senses the obstacle in front of it and including in the software suitable ROS libraries.

However, this thesis details the walking capability of the new robot alone, while its other multifunctional capacities have not been put to test or verified. But, many advanced sensors and highly skilled processing power like image processing has been included enabling further study. Image processing adopted in the future study may include the finding of an item by a robot which is a work done in the machine vision framework. In view of the intention to copy the human vision capacity, a PC vision framework utilizes electronic parts and calculations in a like manner human eyes and mind performs. The Open Source Computer Vision Library (OpenCV) is the most utilized program libraries as a part of

robotics to notice, track and recognize the scene caught by image sensors. OpenCV is a program in python language having numerous libraries for PC visions dreams intended to investigate, prepare and comprehend the items in recordings and photographs meant to deliver data.

Dynamic and stable walking biped humanoid robot is the prima donna of the robot research and industry now. Current humanoid bipedal robots have the capacity only to walk and act in homes, hospitals, and human companion service centers [5]. This problem is being tackled in this research project for the first time by fusing together two different commercially available robots possessing and competently satisfying all the above requirements together.

The future directions of the present work are enlisted below:

- OpenCV, an open source library of image processing along with a web camera can be added to enhance the walking and obstacle avoidance capabilities of the humanoid robot.

- Voice recognition software – PocketSphinx[16][58], can be added and this would increase the responsiveness of the humanoid robot developed here.

Publications:

- Ganesh K. K., Z. Yang, V. Gandhi, T. Geng, "*ROS based autonomous control of a humanoid robot*", International Conference of Artificial Neural Networks and Machine Learning – ICANN, 6-9 September 2016.

- Ganesh K. K., V. Gandhi, Z. Yang, T. Geng, "Using Robot Operating System (ROS) and Single Board Computer to Control Bioloid Robot Motion", 8th Towards Autonomous Robotic Systems (TAROS) Conference, 19-21 July 2017. (Submitted)

[16] PocketSphinx is a simple speech recognition facility, particularly made suitable for handheld and cell phone devices, moreover it works similarly well on the desktop also.

REFERENCES

1. "Robotics - definition of robotics in English | Oxford Dictionary " [Online]. Available: https://en.oxforddictionaries.com/definition/robotics.

2. A. Takanishi, Y. Ogura, and K. Itoh, "Some Issues in Humanoid Robot Design," Springer Tracts in Advanced Robotics Robotics Research, pp. 357–372.

3. A. Gandhi, "Basics of Robotics," Basics of Robotics, 2013. [Online]. Available: http://www.slideshare.net/ameyagandhi/basics-of-robotics.

4. R. Jarvis: Intelligent Robotics: Past, Present and Future. IJCSA 5(3): 23-35 (2008)

5. M. Koga, Y. Hosoda and T. Moriya, 'Humanoid Robots', [Online] www.hitachi.com/ICSFiles/afieldfile/2009/09/14/r 2009_04

6. R. C. Arkin, Behavior-based robotics. Cambridge, MA: MIT Press, 1998. ISBN 0-262-01165-4.

7. J. Y. Kim, III-Woo Park and Jun-Ho oh, 'Design and Walking Control of the Humanoid Robot KHR-2', ICCAS2004, August 25-27, The Shangri-la Hotel, Bangkok, Thailand.

8. Open CV, [Online]. http://opencv.org/

9. Direct Show, Microsoft Windows [Online]. https://msdn.microsoft.com/enus/library/windows/desktop/dd375454(v=vs.85).aspx

10. "Is Arduino suitable to be a Humanoid Robot controller?" microcontroller. [Online]. http://electronics.stackexchange.com/questions/4163/is-arduino-suitable-to-be-a-humanoid-robot-controller.

11. D. Paramkusam, Roboticist, 'What-CPU-is-enough-for-running-a-humanoid-robot' Question and Answers - Written Jul 26, 2015.

12. T. Foote on April 15, 2015 10:55 AM, "Visualizer of delta robots using ROS and EtherCAT," - ROS robotics news. [Online]. Available: http://www.ros.org/news/2015/04/visualizer-of-delta-robots-using-ros-and-ethercat.html.

13. Python SDK"NAO Software 1.14.5 documentation," SDKs —. [Online]. Available: http://doc.aldebaran.com/1-14/dev/sdk.html.

14. K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The Development of Honda Humanoid Robot", in Proc. IEEE Int. Conf. on Robotics and Automations, pp.1321-1326, 1998.

15. H. Lim, Y. Kaneshima and A. Takanishi. Online walking pattern generation for biped humanoid robot with trunk. Proceedings of the IEEE International Conference on Robotics & Automation, pp.3111-3116, 2002.

16. S. Kagami, K. Nishiwaki, J. J. Kuffner, Y. Kuniyoshi, M. Inaba and H. Inoue, 2002. Online 3D vision, motion planning and biped locomotion control coupling system of humanoid robot: H7.

17. D. Wollherr, "*Design and Control Aspects of Humanoid Walking Robots*," Thesis (Dr. -Ing) .Technical University of Munich, 2002.

18. J. Camilo, 'Load Sensors Increase Humanoid Robot's sensitivity', [Online] April 2, 2015. http://www.assemblymag.com/articles/92788-load-sensors-increase-humanoid-robots-sensitivity

19. T. Harris. 'How Robots Work', HowStuffWorks.com. [Online] 16 October 2016. http://www.howstuffworks.com/hsw-contact.htm

20. "Best microcontroller for me to use?," Let's Make Robots!, 04-Aug-2010. [Online]. Available: http://letsmakerobots.com/node/40140.

21. "Motor Controllers," SuperDroid Robots - Home Page. [Online]. http://www.superdroidrobots.com/shop/custom.aspx/motor-controllers/60/.

22. ‘Define operating System. What are the functions and types of operating systems’, [Online], January 09, 2010. http://www.enotes.com/homework-help/define-operating-system-write-down-function-types-128847

23. K. Jackie. "Proposal for Implementation of Real-time Systems in ROS 2". [Online]; http://design.ros2.org/articles/realtime_proposal.html

24. A. Martinez, E. Fernández, A. Romero, and E. Fernandez, Learning ROS for robotics programming: A practical, instructive, and comprehensive guide to introduce yourself to ROS, the top-notch, leading robotics framework. Birmingham: Packt Publishing, 2013.

25. C. Ling, "Robot Operating System Tutorial ROS Basic", Summer School, Shanghai University, July, 23, 2015.

26. "Basic concepts | Erle Robotics Docs", Docs.erlerobotics.com, 2016. [Online].http://docs.erlerobotics.com/robot_operating_system/ros/basic_concepts.

27. M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E.Berger, R. Wheeler, and A. Ng, “ROS: An open-source robot operating system,” in Proc. ICRA Workshop Open Source Software, 2009.

28. N. V. Patel, ‘Robot operating System Is the Future of Software for Autonomous Devices’, [Online]; January 19, 2016 https://www.inverse.com/article/10346-robot-operating-system-is-the-future-of-software-for-autonomous-devices

29. K. P. Berthold Horn, ‘Robot Vision’, MIT Press ISBN 0-262-08159-8

30. M. Brady, J. M. Hollerbach, T. Johnson, T. Lozano-Perez, and M. Mason, ‘Robot Motion: Planning and Control’, MIT Press, ISBN 0-262-12182-X.

31. J. J. Craig, ‘Introduction to Robotics: Mechanics and Control’, Addition Wesley, ISBN 0-201-09528-9.

32. K. Seungsu, K. ChangHwan-Kim, Y. Bumjae, and O. Sangrok, 'Stable Whole-body Motion Generation for Humanoid robots to Imitate Human Motions', 2009 IEEE/RSJ International.

33. S. Wehner, M. Bennewitz, Optimizing the Gait of a Humanoid Robot Towards Human-like Walking, In 4th European Conference on Mobile Robots, Mlini/Dubrovnik, Croatia. 2009

34. A. Shahriara and T. Sariel, 'A Differential Steering System for Humanoid Robots, ECMR, 2011.

35. B. Goodwine and J. Burdick, 'Gait Controlability for legged Robots', Robotics and Automation, 1998.

36. L. Sung-Hee and A. Goswami, 'A Momentum-based Balance Controller for Humanoid robots on Non-level and Non-stationary Ground', Autonomous Robots, 2012 – Springer

37. M. J. Gielniak, C. K. Liu and A. I. Thomaz, 'Generating Human-like Motion for Robots', The International Journal of Robotics Research, 2013.

38. A. Ude, Christopher G, Atkeson and M. Riley, 'Programming Full-Body Movements for Humanoid Robots by observation', Robotics and autonomous systems, 2004 – Elsevier

39. M. Do, D. Gehrig, H. Kohne, P. Azad, P. Pastor, T. Asfour, 'Transfer of Human Movements to Humanoid Robots', Humanoid Robots, 2008, 8th IEEE-RAS International Conference.

40. Y. Davidor. Genetic Algorithms and Robotics: Genetic Algorithms and Robotics - A Heuristic Strategy for Optimism. World Scientific Pub Co, 1991.

41. T. Arakawa and T. Fukuda. Natural motion trajectory generation of biped locomotion robot using genetic algorithm through energy optimization. In Systems, Man, and Cybernetics, 1996. IEEE International Conference on, volume 2, pages 1495 –1500, Oct 1996.

42. D. Schreiner and C. Punzengruber, 'Parametrizing Motion Controllers of Humanoid Robots by Evolution', INFORMATIK 2011 - Informatik schafft Communities

43. M. Vukobratovic, New Frontiers in Robotics: Volume 2 Dynamics and Robust Control of Robot-Environment Interaction, Fraunhofer Institute for Production Systems & Design Technology IPK, Berlin, Germany, – 2009

44. K. JUNG-YUP, P. ILL-WOO and O. JUN,' Experimental realization of dynamic walking of the biped humanoid robot KHR-2 using zero moment point feedback and inertial measurement', Advanced Robotics, Vol. 20, No. 6, pp. 707–736 (2006)

45. W. Park, J. Y. Kim, J. H. Oh, 'Online Biped walking Pattern Generation for Humanoid Robot KHR-3 (KAIST Humanoid Robot-3, IEEE,HUMANOIDS, 2006

46. J. Y. Kim, I. W. Park and J. H. Oh, 'Walking Control Algorithm of Biped Humanoid Robot on Uneven and Inclined Floors', Journal of Intelligent and Ribotic Systems, April 2007, Volume 48, Issue 4 pp 457-484

47. T. Lozano-Pérez and M. A. Wesley, (1979) An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles. Commun. ACM 22 (10): 560-570

48. S. MLaValle, and J. J. Kuffner. (2001) Rapidly-exploring random trees: Progress and prospects. In B. R. Donald, K. M. Lynch, and D. Rus, editors, Algorithmic and Computational Robotics: New Directions, pages 293--308. A K Peters, Wellesley, MA,

49. R. A. Jarvis. (1994) On Distance Transform Based Collision-Free Path Planning for Robot Navigation in Known, Unknown and Time-Varying Environments, invited chapter for a book entitled 'Advanced Mobile Robots' edited by Professor Yuan F. Zang World Scientific Publishing Co. Pty. Lt, pp. 3-31.

50. G. Hoy, 'Robots could cause Australian economy 5 million jobs, experts warn, as companies look to cut jobs', ABC News, Australian Broadcasting Corporation, Retrieved 29 may 2014

51. Telecom glossary "bot" Alliance for Telecommunications solutions, 2001-02-28 Archived from the original 2007-02-02, Retrieved on 2007-09-05

52. R. Grimmett, 'Mastering Beagle Bone Robotics', PACKT Publishing, Mumbai, 2014

53. H. Ayala, Y. Fu, and J. Fu, 'An intial study of Bioloid Humanoid Robot & Beyond', International Conference on Frontiers in education, CS and CE|FECS'! 5|pp17-23.

54. "BIOLOID Premium Robot Kit - RobotShop." [Online]. http://www.robotshop.com/en/robotis-bioloid-premium-robot-kit.html

55. G. Michael, S. Alexey "Bioloid Robot Project", winter 2012 – 2013.

56. G. K. Kalyani, Z. Yang and V. Gandhi, ‘A ROS based Humanoid Robot Structure for Autonomous Control’, ICANN 2016 Scientific Program, BarcelonaTech, Universitat Polit`ecnica de Catalunya Edifici V`ertex, September 6-9, 2016

57. Z. Yang, ‘Dynamic Control of Walking leg Joint: A Building Block Model Perspective’, ICNC 2011, pp 359-463

58. "cmusphinx/pocketsphinx", *GitHub*, 2016. [Online]. Available: https://github.com/cmusphinx/pocketsphinx. [Accessed: 13- Nov- 2016].

59. J. J. Kuffner Jr. and S. M. Lavalle ‘RRT-Connect: An efficient approach to Single-Query Path Planning’, International Conference on Robotics and Automation, San Francisco, CA, April 2000, pp 995-1001.

60. A. A. Saputra and I. A. Sulistijono ‘Biologically Inspired Control System for 3-D Locomotion of a Humanoid Biped Robot’, IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 46, no. 7, July 2016, pp 898 – 911.

61. Z. Yu and W. Zhang, ‘Gait planning of omnidirectional walk on inclined ground for biped robots’, IEEE Transactions on Systems, Man and Cybernetics, Vol. 46, No. 7, July 2016, pp 888 – 897.

62. K. Chang, C. Koh, and C. S. George Lee, ‘An automatic design of factors in a human-pose estimation system using neural networks’, IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 46, No. 7, July 2016, pp 875 – 887.

63. ROSCORE. [Online]. Available: http://wiki.ros.org/roscore, [Accessed: Oct. 2016]

64. ROS TOPIC [Online]. Available: http://wiki.ros.org/Topics, [Accessed: Oct. 2016].

65. Rqt_graph. [Online]. Available: http://wiki.ros.org/rqt_graph, [Accessed: Nov. 2016].

APPENDIX A

DYNAMIC WALKING - ALGORITHM AND CODE

```
#!/usr/bin/env python


import os
import gyro
import dynamixel
import sys
import subprocess
import optparse
import yaml
import time
import serial
import rospy
import Adafruit_BBIO.ADC as ADC
from std_msgs.msg import String

print "Select options "
userinput=raw_input("1.Slow \n2.Normal Walk \nEnter Choice : ")

if userinput == '1':
    dly=1
    spd=5

elif userinput == '2':
    dly=0.3
    spd=700

else:
    print "Not Valid Input !"
    sys.exit()

dynid=""
d3=""
d7=""
d8=""
d9=""
d10=""
d11=""
d12=""
d14=""
d15=""
d16=""
d17=""
d18=""

myActuators = list()

def read_adc():
    ADC.setup()
    value = ADC.read_raw("P9_35")
    values = value/1000
    print values
    return values

def read_lean():
```

```
    xvalue,yvalue = gyro.movement()
    return xvalue,yvalue


def default_Positions():
    global d3,d7,d8,d9,d10,d11,d12,d14,d15,d16,d17,d18,myActuators
    d7.goal_position = 362#352
    d8.goal_position = 672
    d3.goal_position = 512
    d9.goal_position = 512
    d10.goal_position = 512
    d11.goal_position = 512
    d12.goal_position = 512
    d14.goal_position = 512
    d15.goal_position = 512
    d16.goal_position = 512
    d17.goal_position = 512
    d18.goal_position = 512
    print "Initial Position On Sensor"

def main(settings):
    global d3,d7,d8,d9,d10,d11,d12,d14,d15,d16,d17,d18,myActuators
    loop=1
    # Establish a serial connection to the dynamixel network.
    # This usually requires a USB2Dynamixel
    serial =
dynamixel.SerialStream(port=settings['port'],baudrate=settings['baudRate'],
timeout=10)
    # Instantiate our network object
    net = dynamixel.DynamixelNetwork(serial)

    pub = rospy.Publisher('robot', String, queue_size=10)
    rospy.init_node('walk_sensor', anonymous=False, log_level=rospy.INFO,
disable_signals=True)

    # Populate our network with dynamixel objects
    for servoId in settings['servoIds']:
        newDynamixel = dynamixel.Dynamixel(servoId, net)
        net._dynamixel_map[servoId] = newDynamixel
    pub.publish(rospy.get_time)


    if not net.get_dynamixels():
      print 'No Dynamixels Found!'
      sys.exit(0)
    else:
      print "...Done"


    for dyn in net.get_dynamixels():
        myActuators.append(net[dyn.id])

    for actuator in myActuators:       # Set up the servos to Initial
        #dynid=str(actuator).split(" ")[1]
        actuator.moving_speed = 100
        actuator.torque_enable = True
        actuator.synchronized = True
        actuator.torque_limit = 1000
        actuator.max_torque = 1000

    d3 = myActuators[0]
```

```
    d7 = myActuators[1]
    d8 = myActuators[2]
    d9 = myActuators[3]
    d10 = myActuators[4]
    d11 = myActuators[5]
    d12 = myActuators[6]
    d14 = myActuators[7]
    d15 = myActuators[8]
    d16 = myActuators[9]
    d17 = myActuators[10]
    d18 = myActuators[11]

    d7.goal_position = 362#352
    d8.goal_position = 672
    d3.goal_position = 512
    d9.goal_position = 512
    d10.goal_position = 512
    d11.goal_position = 512
    d12.goal_position = 512
    d14.goal_position = 512
    d15.goal_position = 512
    d16.goal_position = 512
    d17.goal_position = 512
    d18.goal_position = 512
    print "Initial Position"
    net.synchronize()   # Send all the commands to the servos.
    #time.sleep(4)
    #for servoId in settings['servoIds']:
    #    print "Current Position of "+str(servoId)+" :
",net.read_register(int(servoId),36,2)
    time.sleep(1)
    right_initial(net)
    time.sleep(0.5)
    print "-----------------------------------------------Initial---
Completed---------------------"
    '''time.sleep(2)
    d11.goal_position =532
    net.synchronize()
    time.sleep(2)
    d11.goal_position =512
    net.synchronize()
    time.sleep(2)
    d11.goal_position =492
    net.synchronize()
    time.sleep(2)'''


    while True:
        left(net)
        time.sleep(dly)
        if read_adc() < 0.5: break

        right(net)
        time.sleep(dly)
        if read_adc() < 0.5: break
        loop+=1
        print "-----------------------------------------------loop----------
-----------"+str(loop)
    default_Positions()
    net.synchronize()
```

```
def right_initial(net):
    global dly,spd
    global d3,d7,d8,d9,d10,d11,d12,d14,d15,d16,d17,d18

    d9.goal_position = 462                          # Step 1 LEFT SIDE LEAN
    d10.goal_position = 462
    d17.goal_position = 462
    d18.goal_position = 462
    print "Step 1 done wait "+ str(dly)
    net.synchronize()   # Send all the commands to the servos.

    time.sleep(dly)
    time.sleep(dly)
    x,y =  read_lean()
    print x,y

    d9.goal_position = 422                           # Step 2 RIGHT LEG
LIFT
    d18.goal_position = 422
    d10.goal_position = 512
    print "Step 2 done wait "+ str(dly)
    net.synchronize()   # Send all the commands to the servos.

    time.sleep(dly)

    d12.goal_position = 512                          # Step 3 MOVE FRONT
    d14.goal_position = 512
    print "Step 3 done wait "+ str(dly)
    net.synchronize()   # Send all the commands to the servos.

    time.sleep(dly)

    d11.goal_position =432                           # Step 4 SWING LEG
    print "Step 4 done wait "+ str(dly)
    net.synchronize()   # Send all the commands to the servos.

    time.sleep(dly)

    d3.goal_position =442                            # Step 5 KNEE DOWN
    print "Step 5 done wait "+ str(dly)
    net.synchronize()   # Send all the commands to the servos.

    time.sleep(dly)

    d9.goal_position =462                            # Step 6 REST POSITION
    d10.goal_position =462
    d17.goal_position = 462
    d18.goal_position =462
    print "Step 6 done wait "+ str(dly)
    net.synchronize()   # Send all the commands to the servos.
    time.sleep(dly)
    #for servoId in settings['servoIds']:
    #    print "Current Position of "+str(servoId)+" :
",net.read_register(int(servoId),36,2)

#====================first_step=========================================#

def left(net):
    global dly,spd
    global d3,d7,d8,d9,d10,d11,d12,d14,d15,d16,d17,d18
```

```
    d9.goal_position = 572  #572                        # Step 1 RIGHT SIDE
LEAN
    d10.goal_position = 572#572
    d17.goal_position = 572#572
    d18.goal_position = 572#572
    #d15.goal_position = 492
    print "Step 1 done wait "+ str(dly)
    net.synchronize()    # Send all the commands to the servos.

    time.sleep(dly)
    time.sleep(dly)
    time.sleep(dly)

    x,y = read_lean()
    print x,y
    if (x>18):
        default_Positions()
        net.synchronize()
        sys.exit()

    d9.goal_position = 512                          # Step 2 LEFT LEG
LIFT////
    d10.goal_position =592
    d17.goal_position =602#592
    d8.goal_position = 672#672
    #d11.goal_position = 512#
    print "Step 2 done wait "+ str(dly)
    net.synchronize()    # Send all the commands to the servos.

    time.sleep(dly)

    d11.goal_position = 512#482                         # Step 3 MOVE FRONT
    d3.goal_position = 512
    d15.goal_position = 512
    print "Step 3 done wait "+ str(dly)
    net.synchronize()    # Send all the commands to the servos.

    time.sleep(dly)
    #time.sleep(dly)

    d12.goal_position = 592                          # Step 4 SWING LEG
    print "Step 4 done wait "+ str(dly)
    net.synchronize()    # Send all the commands to the servos.

    time.sleep(dly)

    d14.goal_position = 582                          # Step 5 KNEE DOWN
    print "Step 5 done wait "+ str(dly)
    net.synchronize()    # Send all the commands to the servos.

    time.sleep(dly)

    d9.goal_position = 572                          # Step 6 REST POSITION
    d10.goal_position = 572
    d17.goal_position = 572
    d18.goal_position = 572
    print "Step 6 done wait "+ str(dly)
    net.synchronize()    # Send all the commands to the servos.

    time.sleep(dly)
    #for servoId in settings['servoIds']:
```

```
    #    print "Current Position of "+str(servoId)+" :
",net.read_register(int(servoId),36,2)
    time.sleep(dly)
#============================step forward============================#
def right(net):
    global dly,spd
    global d3,d7,d8,d9,d10,d11,d12,d14,d15,d16,d17,d18

    d9.goal_position = 462                          # Step 1 LEFT SIDE LEAN
    d10.goal_position =462
    d17.goal_position =462
    d18.goal_position =462
        #else: actuator.goal_position = 512
    print "Step 1 done wait "+ str(dly)
    net.synchronize()   # Send all the commands to the servos.

    time.sleep(dly)
    time.sleep(dly)
    #time.sleep(dly)
    x,y = read_lean()
    print x,y
    if(x<-10):
        default_Positions()
        net.synchronize()
        sys.exit()
    #time.sleep(2)

    d9.goal_position = 422       #432                  # Step 2 RIGHT LEG
LIFT
    d18.goal_position =422#422
    d10.goal_position =512
    print "Step 2 done wait "+ str(dly)
    net.synchronize()   # Send all the commands to the servos.

    time.sleep(dly)
    time.sleep(dly)

    d12.goal_position =512                          # Step 3 COME FRONT
    d14.goal_position =512
    print "Step 3 done wait "+ str(dly)
    net.synchronize()   # Send all the commands to the servos.

    time.sleep(dly)

    d11.goal_position =432                          # Step 4 SWING LEG
    print "Step 4 done wait "+ str(dly)
    net.synchronize()   # Send all the commands to the servos.

    time.sleep(dly)

    d3.goal_position =442                           # Step 5 KNEE DOWN
    print "Step 5 done wait "+ str(dly)
    net.synchronize()   # Send all the commands to the servos.

    time.sleep(dly)

    d9.goal_position =462                           # Step 6 COME TO
POSITION
    d10.goal_position =462
    d17.goal_position =462
    d18.goal_position =462
```

```
    print "Step 6 done wait "+ str(dly)
    net.synchronize()    # Send all the commands to the servos.
    time.sleep(dly)
    #for servoId in settings['servoIds']:
    #    print "Current Position of "+str(servoId)+" :
",net.read_register(int(servoId),36,2)


def validateInput(userInput, rangeMin, rangeMax):
    try:
        inTest = int(userInput)
        if inTest < rangeMin or inTest > rangeMax:
            print "ERROR: Value out of range [" + str(rangeMin) + '-' +
str(rangeMax) + "]"
            return None
    except ValueError:
        print("ERROR: Please enter an integer")
        return None

    return inTest

if __name__ == '__main__':

    parser = optparse.OptionParser()
    parser.add_option("-c", "--clean",
                      action="store_true", dest="clean", default=False,
                      help="Ignore the settings.yaml file if it exists and
\
                      prompt for new settings.")

    (options, args) = parser.parse_args()

    # Look for a settings.yaml file
    settingsFile = 'settings.yaml'
    if not options.clean and os.path.exists(settingsFile):
        with open(settingsFile, 'r') as fh:
            settings = yaml.load(fh)
    # If we were asked to bypass, or don't have settings
    else:
        settings = {}
        if os.name == "posix":
            portPrompt = "Which port corresponds to your USB2Dynamixel? \n"
            # Get a list of ports that mention USB
            try:
                possiblePorts = subprocess.check_output('ls /dev/ | grep -i
usb',
                                                        shell=True).split()
                possiblePorts = ['/dev/' + port for port in possiblePorts]
            except subprocess.CalledProcessError:
                sys.exit("USB2Dynamixel not found. Please connect one.")

            counter = 1
            portCount = len(possiblePorts)
            for port in possiblePorts:
                portPrompt += "\t" + str(counter) + " - " + port + "\n"
                counter += 1
            portPrompt += "Enter Choice: "
            portChoice = None
            while not portChoice:
                portTest = raw_input(portPrompt)
                portTest = validateInput(portTest, 1, portCount)
```

```
                if portTest:
                    portChoice = possiblePorts[portTest - 1]

        else:
            portPrompt = "Please enter the port name to which the
USB2Dynamixel is connected: "
            portChoice = raw_input(portPrompt)

        settings['port'] = portChoice

        # Baud rate
        baudRate = None
        while not baudRate:
            brTest = raw_input("Enter baud rate [Default: 1000000 bps]:")
            if not brTest:
                baudRate = 1000000
            else:
                baudRate = validateInput(brTest, 9600, 1000000)

        settings['baudRate'] = baudRate

        # Save the output settings to a yaml file
        with open(settingsFile, 'w') as fh:
            yaml.dump(settings, fh)
            print("Your settings have been saved to 'settings.yaml'. \nTo "
+
                  "change them in the future either edit that file or run
" +
                  "this example with -c.")


    main(settings)
```